

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Навчально–науковий інститут економіки та управління

Обліково–фінансовий факультет

Кафедра інформаційних систем і технологій



ІНФОРМАТИКА

Методичні рекомендації

до виконання практичних робіт для здобувачів вищої освіти 1 курсу факультету менеджменту, денної форми навчання, освітнього ступеню «бакалавр», напряму підготовки 6.030502 – «Економічна кібернетика»

МИКОЛАЇВ

2016

УДК 330.47
ББК 30.973+65.26
І-74

Друкується за рішенням науково-методичної комісії обліково-фінансового факультету Миколаївського національного аграрного університету від « 20 » квітня 2016 р., протокол № 8.

Укладач:

Ю. В. Волосяк – канд. техн. наук, доцент кафедри інформаційних систем і технологій Миколаївського національного аграрного університету

Рецензенти:

І. П. Атаманюк – д-р техн. наук, доцент кафедри вищої та прикладної математики Миколаївського національного аграрного університету;

Є. Д. Борчик – канд. фіз.-мат. наук, доцент кафедри комп'ютерних систем та мереж МНУ ім. В.О. Сухомлинського

ЗМІСТ

ВСТУП.....	7
ПРАКТИЧНІ РОБОТИ.....	15
ПР №1 Створення листівки	15
ПР №2 Засоби консольного введення/виведення в Delphi.....	19
ПР №3 Створення програми “Обмін валюти”.....	29
ПР №4 Створення програми обміну валюти у двох напрямках	32
ПР №5 Програмування основних циклічних алгоритмів	38
ПР №6 Розв’язування задач з циклічним обчислювальним процесом табулювання функції та з заданим числом повторень	45
ПР №7 Розробка алгоритмів і програм з використанням множин	49
ПР №8 Програмування алгоритмів з використанням одновимірних масивів	52
ПР №9 Програмування алгоритмів з двовимірними масивами	56
ПР №10 Програмування алгоритмів з використанням символічних рядків.....	60
ПР №11 Програмування алгоритмів використанням записів.....	65
ПР №12 Використання компонентів типу TLabel, TEdit та TМето для вводу/виводу даних	71
ПР №13 Програмування алгоритмів з використанням компонентів StringGrid прои написані програми.....	76
ПР №14 Програмування алгоритмів з використанням динамічних структур даних	82
ПР №15 Робота з графікою.....	91
РЕКОМЕНДОВАНА ЛІТЕРАТУРА.....	101

Вступ

Програму Delphi запускають за звичайними правилами операційної системи. Наприклад якщо у загальному вікні Windows на робочому столі є окрема піктограма Delphi, то для запуску програми достатньо вибрати її за допомогою мишки. Інший спосіб – через кнопку Пуск панелі задач та послідовне вибирання елементів спадних меню: Пуск – Програми – Delphi x. Зачекавши декілька секунд, побачимо на екрані чотири головні робочі вікна Delphi (рис. 1.1), які є основою середовища розробки. Зазначимо що Delphi, як і будь-яка інша програма в середовищі Windows, може працювати з іншими програмами, до яких можна переходити під час роботи. Отже середовище Delphi призначене для розробки прикладних програм під Windows на базі мови Object Pascal.

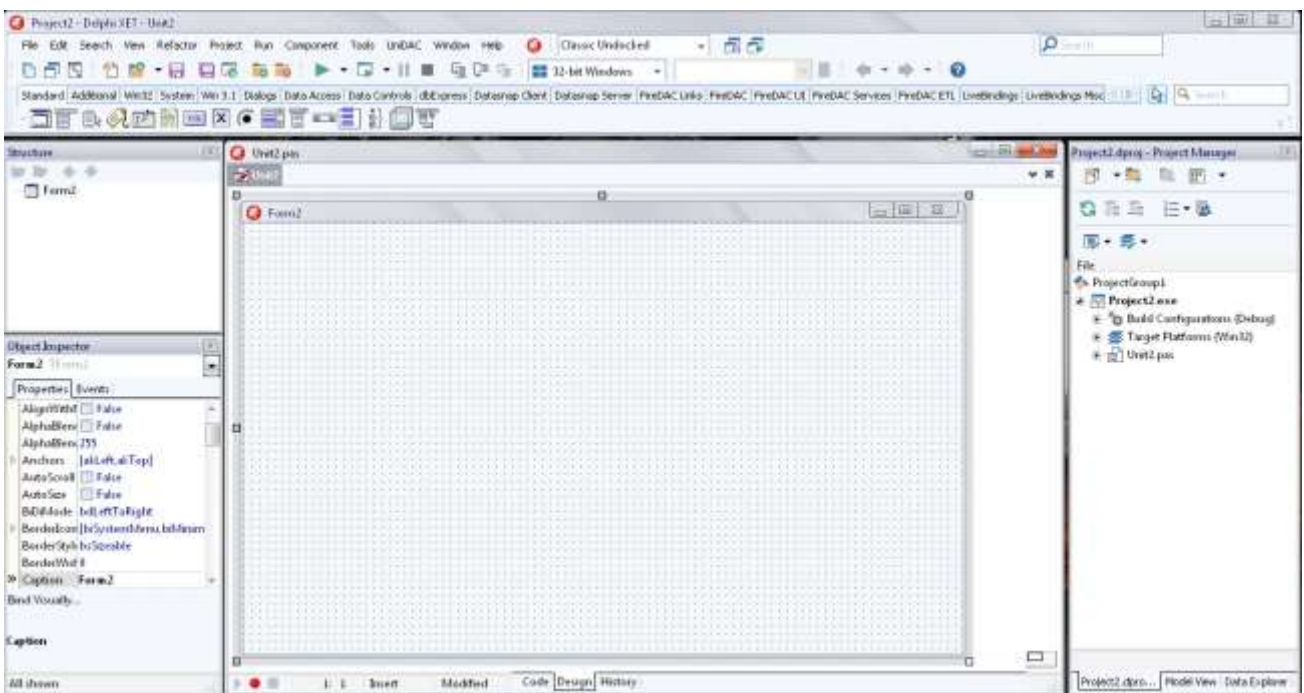


Рис. 1.1 Робочі вікна Delphi після завантаження.

Середовище складається з чотирьох вікон, якими можна керувати як багатовіконною прикладною програмою з інтерфейсом одного документа (single document interface – SDI). Це означає що розмірами та розташуванням кожного вікна можна керувати незалежно.

Головне вікно розташоване зверху і центральним елементом керування під час роботи в Delphi. Воно складається з трьох виділених елементів: рядка меню, панелі швидкого доступу, панелі компонентів.

Вікно інспектора об'єктів розміщене ліворуч знизу екрана. Інспектор об'єктів складається з двох сторінок – *властивості*

(*Properties*) та *події (Events)*). За допомогою інспектора об'єктів задають параметри та поведінку окремих елементів (компонент) з яких будують програму.

Вікно форми (праворуч під головним вікном) є робочою ділянкою для розташування елементів керування програмою під час її майбутнього використання. Складніші програми потребують кількох різних форм, кожен з яких незалежно проектується у Delphi в рамках того ж проекту. Видимою і доступною для проектування в кожен момент часу можна зробити будь-яку з визначених форм.

Вікно редактора коду дає змогу вводити та редагувати тексти програм мовою Object Pascal. Воно розташоване праворуч внизу, однак часто доводиться збільшувати його розміри, щоб одночасно бачити значну частину програми. Вікно редактора коду є багатосторінковим, на кожній сторінці можна редагувати інший текстовий файл (модуль) програми.

Вікна форми та редактора коду під час початкового запуску Delphi можуть частково перекриватися, тому перед початком роботи потрібно за допомогою мишки налаштувати розміри та місце таких вікон на екрані. Форм (вікон) може бути і декілька, тому для їхнього перемикавання використовують команди середовища або мінімізують непотрібні в конкретний момент.

Після запуску Delphi вибирають один з двох варіантів подальшого налаштування. Для створення нової програми доцільно вибрати через меню Delphi команду File – New Application, після чого Delphi перемалює заново вікна форми та редактора коду, і прив'язати нову програму до окремої папки. Якщо ж роботу над програмою розпочато раніше, то потрібно відкрити її командою File – Open або File – Reopen. Обидва варіанти розглянемо детальніше.

1.2 База прикладної програми

Усі прикладні програми, побудовані в Delphi, мають інтерфейс з користувачем на основі вікон. Вікно – це своєрідна база для зовнішнього оформлення програми, яку можна сконструювати та оформити з використанням елементів Delphi, таких як панелі, кнопки, фігури, текстові та редаговані поля, списки рядків, таблиці, смуги перегляду, стандартні діалогові вікна тощо, які називають *компонентами*. На початку роботи над новою програмою вікно форми є порожнім, однак воно має загальні властивості вікон Windows. Щоб переконатися в цьому, зробимо таке. Відразу після запуску Delphi або



переходу до нової розробки виберемо зверху в меню послідовно пункти Tools – Environment Options... – закладки Preferences, і за допомогою мишки поставити позначку V в поля Show compiler progress та Minimize on run (якщо таких ще немає). Після цього натиснемо кнопку ОК. Таким чином ми підготували середовище для зручного спостереження за роботою поки що не існуючої програми, яка складається з порожньої форми. Тепер мишкою натискаємо на кнопку розміщену зверху приблизно під пунктом меню View, якщо зіставити з рис 1.1, і даємо команду Delphi виконати “порожню” програму.

Спочатку побачимо посередині вікно з повідомленнями про хід трансляції програми (рис. 1.2). Після закінчення трансляції відмі вікна середовища Delphi тимчасово зникнуть з екрана, і з'явиться порожнє вікно програми (рис.1.3)

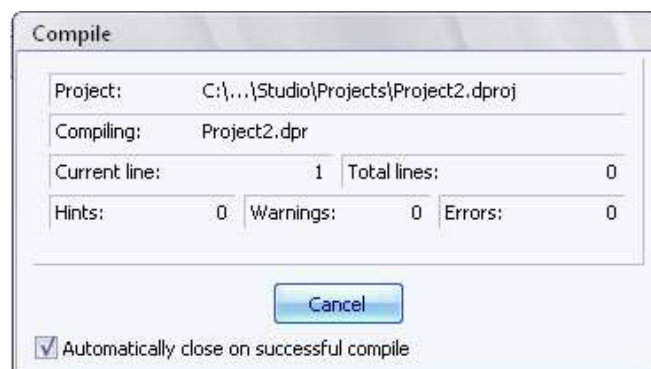


Рис. 1.2 Повідомлення про компіляцію програми



Рис. 1.3 Початкове вікно програми

Таке вікно може виконувати головні команди щодо свого розташування та розміру. Ці команди є стандартними для Windows. Спробуйте за допомогою мишки “схопити” за верхню смугу та перенести вікно в іншу позицію екрана. Після цього натисніть мишкою

на кнопки максимізації, мінімізації та відновлення, які є праворуч вгорі форми. Як бачимо порожнє вікно виконує відповідні команди.

Нарешті натиснемо на кнопку закриття форми. Вікно зникає і ми повертаємось в середовище Delphi, яке відновлює зображення своїх вікон.

Delphi дає стартову базу (вікно та відповідну програму) для правильної роботи в операційній системі Windows, наділяючи базу стандартно прийнятими в Windows, елементами керування та поведінки. Ще не написавши жодного рядка тексту програми, ми можемо спостерігати та контролювати виконання майбутньої програми. На початківців велике враження справляє такий простий спосіб початкового проектування, а на досвідчених програмістів – уява про обсяг програмування, який довелося б виконувати самотійно без засобів Delphi.

1.3 Головні прийоми візуального програмування

1.3.1 Загальне налаштування Delphi

Для побудови програми в Delphi широко використовують засоби, які надає система. Є дві частини побудови: перша- проектування інтерфейсу з використанням стандартних елементів (компонент) та маніпулювання їхніми розмірами й розташуванням; друга - написання фрагментів програмного коду для виконання завдання. Delphi самотійно записує деякі частин програми без зовнішнього втручання, розробникові треба кодувати лише суто свою задачу). Крім того, Delphi формує для майбутньої програми погрібну інформацію у файлах. На всіх етапах розробки програми можна бачити її інтерфейс, перевіряти програму шляхом виконання, змінювати властивості компонент, на яких побудована програма, вилучати чи додавати візуальні компоненти.

На початку роботи з системою Delphi доцільно виконати її мінімальне налаштування.

По-перше, треба підготувати папку (каталог), де будуть зберігатися всі файли майбутньої програми. Це ліпше робити до запуску Delphi.

Нову папку можна створити за звичайними правилами операційної системи Windows. Наприклад, запустити інструмент Мій комп'ютер (My Computer), розташований на робочому столі системи, відшукати потрібний диск та вже наявну папку, відкрити її, після чого через меню вибрати команди Файл – Створити - Папка. Ввівши з клавіатури

потрібне ім'я папки та натиснувши на клавішу Enter, матимемо готову порожню папку.

Свою папку можна створити і пізніше, під час побудови програми. У разі першого запам'ятовування відшукати у стандартному вікні запам'ятовування файлів потрібні диск та папку і за допомогою кнопки вікна Створення – нової папки так само, як і в попередньому випадку, ввести з клавіатури ім'я папки.

По-друге, потрібно вибрати і налаштувати деякі важливі параметри **Delphi**, які постійно впливатимуть на подальшу роботу. Вище зазначено, що в меню треба послідовно вибрати пункти **Tools – Environment Options...** – закладки **Preferences** і за допомогою мишки поставити позначку *V* в поля **Show compiler progress** та **Minimize on run**. Таку ж позначку потрібно поставити в полі **Break on exception**. Для того, щоб можна було записувати в програмі коментарі та літерні рядки українською мовою, треба за такими ж пунктами вибрати закладку **Display** і в рядку **Editor Font** зі спадного списку шрифт *Courier New Cyr*.

Редактор коду розташований в окремому вікні, яке можна закривати та відкривати незалежно від головного та інших вікон (рис.1.4). У цьому вікні відображають та редагують тексти програм.

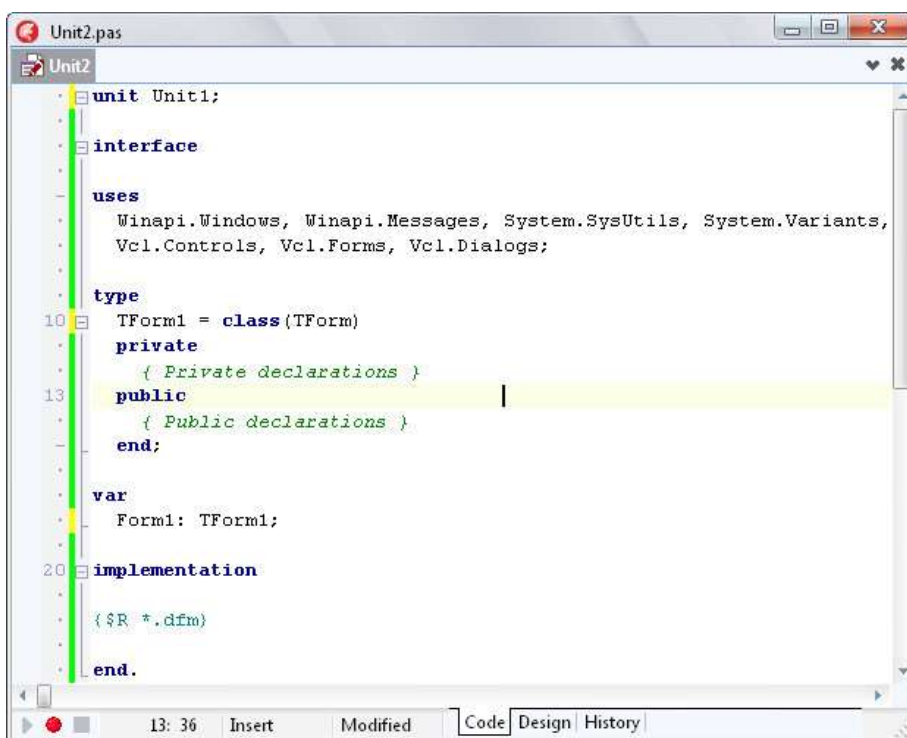


Рис. 1.4 Вікно редактора коду

Праворуч та знизу від тексту є стандартні смуги його перегляду. Зверху над текстом зображені закладки для позначення відкритих

файлів. Кожна закладка містить ім'я відповідного модуля. Для перемикання вікна на потрібний модулі достатньо клацнути мишкою на його закладці (на рис. 1.4 – Unit 1 та Project 1).

У нижній частині вікна є рядок стану. Він містить інформацію про місце курсора на активній сторінці (перше поле ліворуч), друге поле відображає, чи був змінений текст у цьому вікні з моменту останнього запам'ятовування, третє поле повідомляє про режим набору символів на клавіатурі: Insert – уставляння, Overwrite – заміна, Read Only – текст не можна змінювати. Перемикання між режимами Insert та Overwrite виконують клавішею Ins.

Крім того, рядок стану призначений для виведення повідомлень компілятора про помилки в програмі, а також для введення тексту під час виконання команди Search – Incremental Search.

Смугу ліворуч від тексту використовують для виокремлення рядків, на яких будуть розташовані місця зупинки програми. Для цього достатньо клацнути мишкою на смузі навпроти потрібного рядка. Повторне клацання мишкою знімає місце зупинки.

Редактор коду може працювати спільно з налагоджувачем Delphi. Налгоджувач дає змогу трасувати програму безпосередньо у вікні редактора і відобразити в ньому всю поточну інформацію.

Редактор коду Delphi виконує значну частину роботи програміста. Зокрема, під час перенесення у вікно форми нових компонент у клас форми автоматично додає відповідні поля, а в проект - відповідні модулі. У випадку, коли до компоненти додають програму опрацювання події, у вікні редактора коду з'являється базовий (початковий) текст цієї програми, а курсор розташовується на місці майбутнього першого оператора.

Зауважимо, що редактор коду можна використовувати для редагування будь-яких текстових файлів, наприклад, файлів вхідних даних програми, і навіть файлів, не пов'язаних з цією програмою.

1.3.3 Вікно форми

Вікно форми в Delphi можна трактувати як своєрідний будівельний майданчик, на якому відповідно до задуманого плану розташовують компоненти майбутньої програми (рис.1.5).

Форма – це вікно Windows, побудоване одним із допустимих стилів. Зверху над вікном є смуга з заголовком і стандартними кнопками керування вікном. Решта простору вікна форми є робочою ділянкою, що покрита крапковою сіткою згідно з параметрами

сторінки Preferences ознак середовища. Сітка призначена для вирівнювання компонент під час їх розміщування на площині форми.

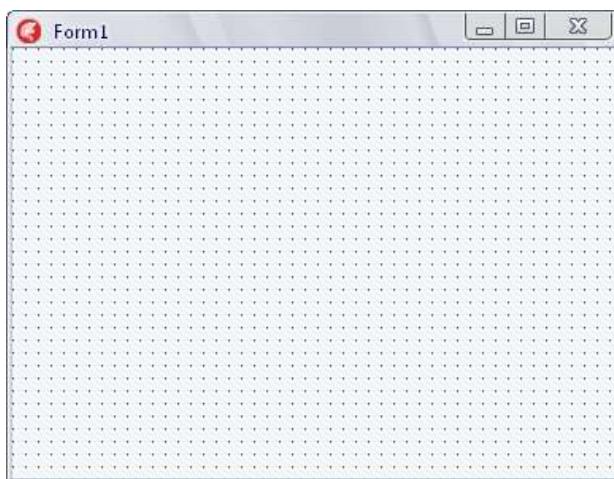


Рис. 1.5. Початкове вікно форми програми

Водночас зі створенням нового проекту автоматично створюється нова форма, яка з'являтиметься під час виконання готової програми. Таку форму називають *головною формою проекту*. Якщо ж проект потребує декількох форм, то нову стандартну форму можна додати командою **File – New Form**, а шаблон форми вибрати з набору в діалоговому вікні командою **File – New – Forms**. Шаблон містить готовий набір елементів керування для типових випадків розробки проектів. **Delphi** дає змогу створювані власні шаблони та зберігати їх разом зі стандартними.

Розглянемо головні етапи роботи з компонентами в робочій ділянці вікна форми. Активна в конкретний момент компонента (та, що має фокус) виділена контурною рамкою з маленьких чорних квадратів по периметру. Фокус між компонентами переміщують мишкою або клавішею Tab.

Декілька компонент, розташованих на формі, можна об'єднати в групу. Для цього треба натиснути на ліву клавішу мишки і пересунути її так, щоб ділянка, обмежена штриховою лінією, що тягнеться за мишкою, захопила потрібні компоненти. Після відпускання клавіші мишки всі об'єднані в групу компоненти будуть виділені контурною рамкою кожен, однак сірого кольору, сама рамка зафіксована лише по кутках кожної компонент. Щоб відмінити об'єднання компонент, досить клацнути мишкою в будь-якому вільному від компонент місці робочої ділянки вікна форми.

Для вилучення компоненти з форми потрібно перевести на неї фокус і натиснути на клавішу Del або вибрати з головного меню команду **Edit – Delete**. Подібно можна вилучити відразу всі компоненти, **об'єднані** попередньо в групу. Вилучену компоненту або групу відновлюють командою **Edit – Undelete**.

У межах робочої ділянки вікна пересувати компоненту найзручніше мишкою, натискаючи на ліву клавішу. Для точного розташування компонент у вікні використовують комбінацію клавіш керування курсором та клавіші **Ctrl**. Змінити розташування компоненти можна також шляхом задання відповідних координат її лівого верхнього кута в інспекторі об'єктів (властивості **Left, Top**).

Розмір компоненти найзручніше змінювати також мишкою. Для цього спершу компоненту роблять активною. Далі пересувають мишку на один з маркерних чорних квадратів, розташованих по периметру. Курсор набуває вигляду двонапрямленої стрілки. Зафіксувавши ліву клавішу, пересувають мишку в потрібному напрямі і відпускають клавішу. Точну зміну розмірів виконують комбінацією клавіш керування курсором та клавіші Shift. Крім того, розміри можна визначити відповідними властивостями (**Height, Width**) в інспекторі об'єктів.

Розміри та розташування всіх компонент у вікні можна одночасно і пропорційно змінити. Для цього використовують команду головного меню **Edit – Scale**, зазначивши зміну масштабу у відсотках.

Якщо у вікні форми вже розташовані декілька компонент, то їх можна вирівнювати як щодо вікна форми, так і одна щодо одної. Для цього застосовують команду головного меню **Edit – Align** або палітру вирівнювання (рис. 1.6), яку викликають командою **View – Alignment Palette**. Ця палітра є окремим вікном, яке можна розташувати на екрані в зручному місці на тривалий час. Вирівнювання виконують для окремих компонент, а також для груп.

Над компонентами та їхніми групами можна виконувати операції вирізання, копіювання в буфер обміну, вставлення з буфера обміну. Однак копіювання та вирізання можливе лише для таких груп компонент, які мають спільного "родича" (форму або компоненту-контейнер).

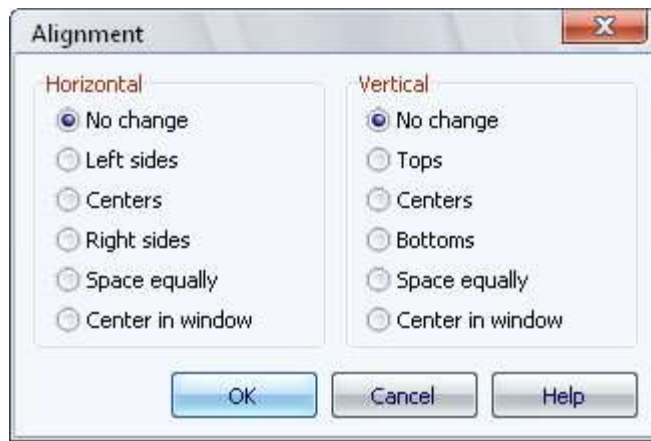


Рис. 1.6 Палітра вирівнювання

Інспектор об'єктів є інструментом середовища розробки Delphi, який дає змогу повністю визначати вигляд та поведінку компонент, що розташовані на формах проекту, а також самої форми. За його допомогою можна задавати потрібні значення властивостей компонент (об'єктів), а також реакцію на стандартні події. Інспектор об'єктів розташований в окремому вікні, яке створюється автоматично під час завантаження Delphi.

Вікно інспектора об'єктів має список компонент поточної активної форми, а також дві сторінки: властивостей **Properties** (рис. 1.7) та подій **Events** (рис. 1.8). Дані з'являються на сторінках інспектора об'єктів лише за наявності відкритого файлу з текстом хоча б одного модуля в редакторі коду.



Рис. 1.7. Сторінка властивостей інспектора об'єктів

У верхній частині інспектора розміщений список компонент в алфавітному порядку, які розташовані на активній формі, включаючи саму форму. Побачити список можна, натиснувши на кнопку списку компонент. Якщо вибрати яку-небудь компоненту із списку мишкою, то компонента стає активною, а обидві сторінки інспектора об'єктів заповняться значеннями її опублікованих властивостей та іменами процедур опрацювання подій, у вікні форми відповідна компонента буде виокремлена контурною рамкою. Якщо в списку нема імені, то це означає, що сторінки інспектора об'єктів містять властивості та методи опрацювання подій групи об'єднаних компонент.

Видиме ім'я активної компоненти побудоване з двох частин: власне імені компоненти та імені типу (класу), до якого вона належить.

Сторінка властивостей складається з двох стовпців: лівий з назвами імен властивостей активної компоненти, а правий з їхніми значеннями. Для поточної властивості в правому стовпці стає акційним поле редактора, вигляд якого залежить від конкретної властивості. Якщо поле редактора не містить ніяких позначок, то це властивість, значення якої треба ввести з клавіатури. Якщо ж поле має кнопку значень переліченого типу (зі стрілкою вниз), то значення можна вибрати зі списку допустимих наперед визначених значень. Побачити список можна, якщо натиснути на цю кнопку. Деякі властивості (наприклад, Font або Icon) для вибору значень мають діалогове вікно. В полі редактора властивостей у цьому випадку видно кнопку з трьома крапками. Після натискання на неї з'явиться діалогове вікно, в якому треба задати по черзі декілька різних значень, що стосуються тієї властивості, а потім його закрити.

Властивості можуть бути простими та комплексними.

Комплексні – це властивості, які складаються з набору інших властивостей, для кожної з яких задано свої значення, їх позначають знаком «+». Набір властивостей відповідно може складатися як з простих, так і з комплексних. Для перегляду набору властивостей потрібно двічі клацнути мишкою на імені комплексної властивості, після чого в стовпці імен властивостей з'явиться додатковий список, а знак перед іменем зміниться на «-». Щоб закрити додатковий список, треба двічі клацнути мишкою на імені комплексної властивості.

Сторінка подій (рис.1.8) також має два стовпці. Лівий – з іменами стандартних подій, на які може реагувати компонента, а правий – з іменами процедур (методів) опрацювання події, які реалізують реакцію компоненти.



Рис. 1.8. Сторінка подій інспектора об'єктів.

Стандартні події виникають під час створення та знищення компоненти, зміни її видимості, натискання на клавіші клавіатури, клацання мишкою тощо. Багато компонент можуть реагувати на специфічні, характерні лише для них події. Кожній стандартній події відповідає фіксоване ім'я методу (процедури) опрацювання.

Початкове правий стовпець є порожній, тобто компонента не реагує на жодні події. Для визначення реакції компонента на подію потрібно перевести курсор мишки на поле редактора події в правому стовпці та двічі клацнути лівою клавішею мишки. Після цього в стовпці імен з'явиться ім'я методу опрацювання, а до вікна відповідного модуля в редакторі коду автоматично додається базовий (початковий) текст цієї програми, і курсор розташується на місці майбутнього першого оператора.

Практична робота №1.

Тема: Створення листівки.

Мета: Вивчення середовища візуального програмування та етапів створення програми в Delphi x.x.

Теоретичні відомості.

Форма (form). Форма володіє властивостями вікна Windows. На формі розташовують усі інші компоненти (елементи керування), такі як кнопки, текстові поля, малюнки, списки тощо. Форма має власний набір властивостей: заголовок (Caption), назву (Name), колір (Color), розміри (Height – висота, Width – ширина), відступ від лівої межі екрана до форми (Left), відступ від верхньої межі (Top), шрифт (Font), BorderStyle – можливість змінювати чи не змінювати розміри вікна форми на етапі виконання програми-проекту тощо. Розміри компонента і відступи задаються пікселями.

Заголовок (Caption) для форми задає користувач відповідно до змісту задачі. Він відображається у рядку заголовка вікна. Заголовок не слід плутати з назвою (Name). Назва – це системна назва об'єкта. Її можна міняти, але це робити не рекомендується. Колір форми (Color) користувач добирає за допомогою інспектора об'єктів із запропонованого списку кольорів. З кольорами можна експериментувати. З шрифтами також варто експериментувати, щоб підібрати шрифт, його розмір та колір до вподоби.

Текстове поле (*label*) – це об'єкт, за допомогою якого на форму можна нанести різноманітні написи, тексти. Він має такі властивості: заголовок (Caption), назву (Name), колір (Color), розмір (Height, Width), доступність (Enabled), видимість (Visible) тощо. Кожний новий об'єкт типу label матиме свою системну назву: label1, label2, label3 і т.д. Доступність (Enabled) може набувати значення true або false. Значення false робить чорний текст сірим. Цікавою властивістю є видимість (Visible). Її значення false робитиме поле невидимим.

Картинка (*image*) – компонент, призначений для вставляння картинок у форму. Окрім перерахованих вище властивостей він має ще властивість Picture, яка дає змогу вибрати у файловій системі деякий графічний файл (наприклад, у форматі *bmp*) і відобразити його на формі. Важливою властивістю картинки є Stretch. Її значення true забезпечує заповнення ділянки потрібного розміру відповідним зображенням.

Значення деяких властивостей можна ввести з клавіатури, інші

значення - вибрати зі списку запропонованих значень, наприклад, для властивості шрифт. Для того, щоб задати потрібний шрифт, його розмір та вигляд написання потрібно активізувати властивість Font у вікні інспектора об'єктів, клікнути лівою клавішею миші на ... (трьох крапках) і в діалоговому вікні, що з'явиться, задати потрібні властивості.

Для об'єкта-форми можна задавати стандартні події. Для цього потрібно виконати такий алгоритм: вибрати об'єкт, перейти на закладку Events у вікні інспектора об'єктів, вибрати одну зі стандартних подій (OnClick – у випадку клацання мишею на формі під час виконання програми, OnClose – у випадку закривання форми, OnDbClick – у випадку подвійного клацання лівою клавішею миші на формі тощо), двічі клацнути на назві події - відкриється програмний модуль з заготовкою відповідної процедури, яку необхідно заповнити командами згідно з метою роботи.

Для об'єктів текстове поле, картинка та інших також можна задати стандартні події OnClick, OnDbClick тощо і запрограмувати їх відповідним чином.

Хід роботи


1. Завантажити Delphi.
2. Задайте властивості форми. Для цього активізуйте (виберіть) Form1 і у вікні Object Inspector уведіть такі значення властивостей:


Properties	Значення
Caption (Заголовок форми)	Листівка
Height (Висота форми)	350
Width (Ширина форми)	500
Color (Колір форми)	на свій розсуд
BorderStyle (Зміна розмірів)	BsDialog (розміри міняти не можна буде)


3. Створіть на диску свою папку. Збережіть створену програму у папку. Вам потрібно буде зберегти два файли: файл програмного модуля (unit1.pas) і файл проекту (project1.dpr). Для цього виберіть команду Save All (Зберегти все) головного меню File. У вікні Save Unit1 As у полі Save in: (Зберегти в:) виберіть робочий диск, після чого виберіть власну папку. У полі File name: задайте назву програмного модуля, якщо вас не влаштовує unit1.pas. Клацніть на кнопці Save. Далі у вікні Save Project1 As виберіть свою папку, і в поле File name: уведіть власну назву проекту. Назви модуля та

проекту мають бути різними.

4. Запустіть проект на виконання. Запустити проект можна різними способами, а саме:

- а) у головному меню вибираємо команду Run → Run;
- б) клацаємо на кнопці Run  на панелі інструментів;
- в) натискаємо на клавішу F9;

Розгляньте вікно вашої Delphi-програми і закрийте його .

5. Вставте у форму текстові поля (об'єкти типу Label) як показано на рис.1 і надайте текстовим полям Label1-Label5 відповідні властивості. Для цього потрібно клікнути мишею на піктограмі Label (рис.3,  – ця кнопка третя на закладці Standard палітри компонентів). Після цього клікнути мишею у тому місці на формі, де має знаходитись текстове поле. У вікні Object Inspector активізуйте Label1 і задайте значення його властивостей. Подібним способом вставте ще чотири текстові поля Label2 та Label3 і задайте значення властивостей Caption, Font, Font style, Size, Color (колір виберіть на свій розсуд):

Поле	Caption	Font style	Size
Label1	Обережно!	Bold	16
Label2	Не губіть живу природу!	так і залишити	14
Label3	WWF FOND	Italic	14

Розташуйте поля якнайкраще на формі методом перетягування об'єктів чи задання властивостей Left, Top тощо. Якщо випадково закриєте вікно форми, то натисніть на кнопку F12.

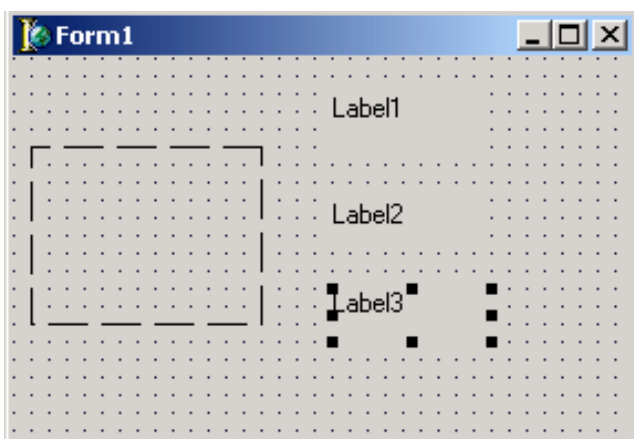


Рис. 1.1

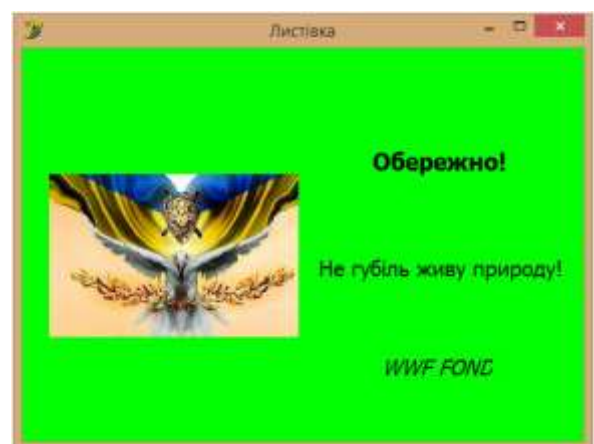



Рис. 1.2



Рис. 1.3 Панель компонентів

6. Збережіть роботу на диску.

7. Вставте картинку як показано на рис. 1. Для цього виберіть закладку Additional (додаткові) палітри компонентів і клацніть мишею на піктограмі Image . У потрібному місці на формі обведіть контур (ділянку) майбутньої картини (Properties у вікні Object Inspector встановіть Width = 260, High = 225). Активізуйте об'єкт Image1 у вікні Object Inspector і виберіть властивість Picture. Клацніть на кнопці з трьома крапками. У діалоговому вікні Picture Editor клацніть на кнопці Load..., знайдіть на диску файл Sample.jpg і клацніть послідовно на кнопках Open, Ok. Щоб зображення заповнило увесь контур. Картинка завелика, тому встановіть у вікні Object Inspector → Stretch як true.

8. Збережіть створену форму (проект) на диску.

9. Запустіть програму-проект на виконання (див. пункт 4). Результат → рис.1.2.

10. Створіть exe-файл програми. Виберіть і виконайте команди головного меню Project → Build All (Сконструювати Все). Виконуваний файл буде називатися Project1.exe.

11. Знайдіть цей файл на диску і дослідіть, який його обсяг.

12. Продемонструйте створену форму викладачу, виконавши exe-файл.

Вимоги до оформлення звіту :

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке форма, властивості форми?
2. Що таке текстове поле, його властивості?
3. Як створити на диску свою папку?
4. Як запустити проект на виконання?
5. Як створити exe-файл програми?
6. Що таке картинка, її властивості?

Практична робота №2 .

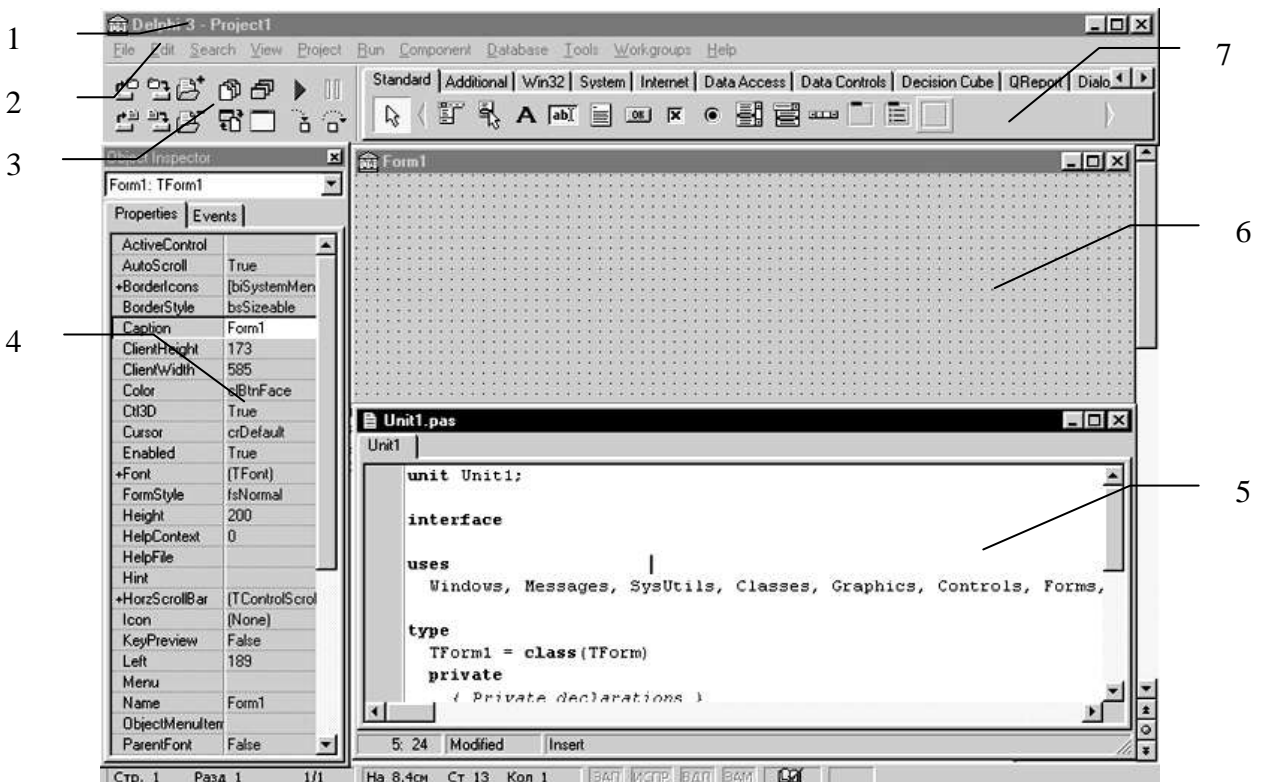
Тема: Засоби консольного введення/виведення в Delphi x.x_

Мета: вивчити основні елементи візуального середовища, освоїти використання найпростіших компонентів DELPHI для введення/виведення консольних даних.

Теоретичні відомості:

1. Завантажте систему візуального програмування DELPHI.

При запуску DELPHI на екрані з'являється панель інтерфейсу. Середовище DELPHI візуально реалізується у вигляді декількох одночасно розкритих на екрані монітора вікон. Кількість, розташування, розмір і вид вікон може змінюватися програмістом.



- 1 – Головне вікно; 2 – головне меню, 3 – піктограми головного меню,
4 - вікно Інспектора Об'єктів; 5 – вікно Редактора Кода,
6 - вікно пустої Форми; 7 – Палитра Компонентів.

Рис. 1.1

Головне вікно завжди присутнє на екрані і призначене для управління процесом створення додатку.

Головне меню має всі необхідні засоби для управління проектом.

Піктограми головного меню полегшують доступ до найбільш часто вживаних команд.

Палітра Компонентів забезпечує доступ до набору бібліотечних програм середовища DELPHI, які описують деякий елемент (компонент), поміщений програмістом у вікно *Форми*. Кожний компонент має певний набір властивостей, які програміст може вибирати і змінювати на свій розсуд. Наприклад, заголовок вікна, напис на кнопці, розмір, колір і тип шрифту і ін. Властивості компонентів приведені в *HELP*.

Вікно Інспектора Об'єктів призначено для зміни властивостей вибраних компонентів і складається з двох сторінок. Сторінка *Properties* (Властивості) призначена для зміни необхідних властивостей компоненту. Сторінка *Events* (Події) – для визначення реакції компоненту на ту або іншу подію (наприклад, клацання кнопки “миші”).

Вікно Форми є інтерфейсом проєктованого Windows-додатку. В це вікно на етапі проєктування додатку поміщаються необхідні компоненти, які розробник бере з *Палітри Компонентів*. Кожній *Формі* проєкту відповідає модуль (**Unit**), текст якого на мові Object Pascal розміщується у вікні *Редактора Коду*.

Вікно Редактора Коду призначено для перегляду, створення і редагування текстів модулів проєкту. При первинному завантаженні у вікні *Редактора Коду* знаходиться текст модуля, що містить мінімальний набір операторів для нормального функціонування порожньої *Форми* як Windows-додаток. При розміщенні деякого компоненту у вікні *Форми*, текст модуля автоматично доповнюється необхідними операторами.

Про всі події, що відбуваються в системі, таких як створення *Форми*, натиснення кнопки миші або клавіатури і т.д., ядро Windows інформує вікна шляхом посилки відповідних повідомлень. Візуальне середовище DELPHI приймає і обробляє повідомлення за допомогою обробників подій (наприклад, клацання кнопки “миші” – подія *OnClick*, створення *Форми* – *OnCreate*).

Для створення обробника події програмісту необхідно розкрити список компонентів у верхній частині вікна *Інспектора Об'єктів* і вибрати необхідний компонент. Потім, на сторінці *Events Інспектора Об'єктів*, натисненням лівої клавіші миші вибрати назву обробника і двічі клацнути по його правій (білої) частині. У відповідь DELPHI активізує вікно *Редактора коду модуля* і покаже заготовку процедури обробки вибраної події. Для кожної оброблюваної події в тексті модуля організовується процедура (*procedure*), між ключовими

словами *begin i end* якої програміст на мові Object Pascal записує необхідний алгоритм обробки події.

Таблиця 1.1

Подія	Опис події
OnActivate	Виникає при активізації Форми
OnCreate	Виникає при створенні Форми. В обробнику даної події слід задавати дії, які повинні відбуватися у момент створення Форми, наприклад установка початкових значень.
OnClick	Виникає при натисненні кнопки миші в області компоненту.
OnDblClick	Виникає при подвійному натисненні кнопки миші в області компоненту
OnKeyPress	Виникає при натисненні клавіші на клавіатурі. Параметр Key має тип Char і містить ASCII-код натискаючої клавіші (клавіша Enter клавіатури має код #13, клавіша Esc – #27 і т.д.). Звичайно ця подія використовується у тому випадку, коли необхідна реакція на натиснення однієї з клавіш.
OnKeyDown	Виникає при натисненні клавіші на клавіатурі. Обробник цієї події одержує інформацію про натискуючу клавішу і натискання клавіш Shift, Alt і Ctrl, а також про натискуючу кнопку миші.

2. Приклад створення консольного додатку.

Завдання:

Створити Windows-додаток для обчислення виразу

$$u = tg^5(\sqrt{x} - y^3) + e^{y/z} \cdot \sin z^2.$$

Чисельні значення даних x , y і z занести з клавіатури у відповідні поля панелі інтерфейсу. Один з можливих варіантів панелі інтерфейсу створюваного консольного додатку показаний на рис. 1.2.

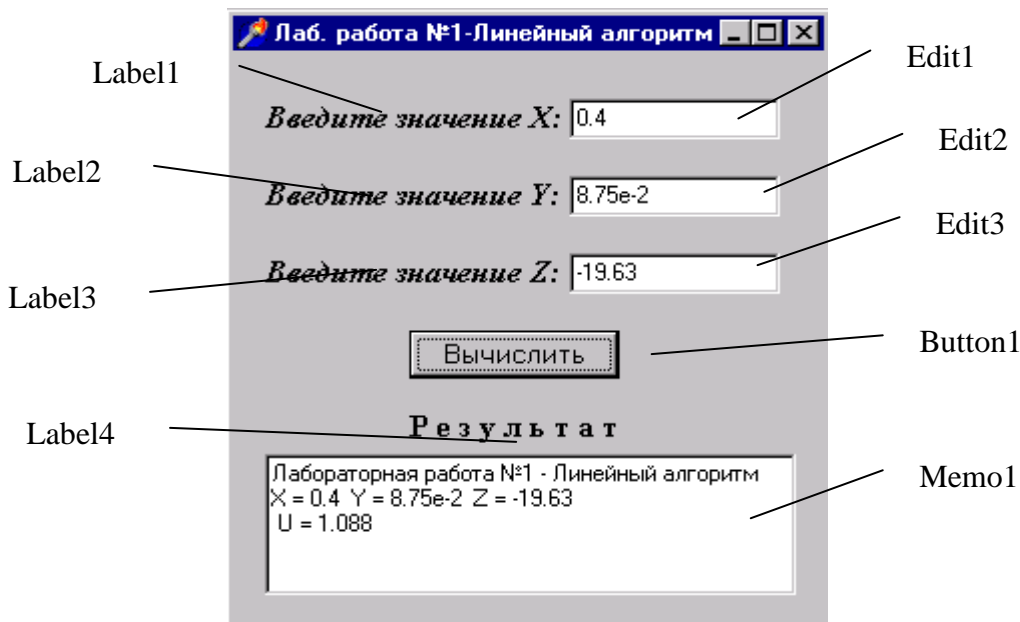



Рис. 1.2

3. Збереження проекту.

У процесі проектування консольного додатку Delphi створює декілька файлів – *проект*. Кожний проект доцільно зберегти в окремій, наперед створеній папці. За допомогою відповідного консольного додатку Windows створимо папку і назвемо її, наприклад, LAB2.





Для збереження проекту відкрийте в головному меню пункт **File** і клацніть “мишею” на опції Save Project As.(Зберегти проект як.). Спочатку Delphi відкриє панель діалогу Save Unit1 As (Зберегти модуль як) для збереження модуля проекту. В цій панелі знайдемо створену папку LAB2 і збережемо в ній модуль під ім'ям, наприклад, UnLinAlg. Зверніть увагу на те, що Delphi за замовчанням привласнить цьому файлу тип *Delphi unit* з розширенням *.pas. Потім відкриється панель діалогу Save Project1 As . Назвемо наш проект, наприклад, PrLinAlg і збережемо його в цій же папці. Тут Delphi дасть файлу тип Delphi project і розширення *.dpr. Переконайтеся в тому, що головне вікно Delphi тепер називається PrLinAlg, вікно головного файлу проекту – PrLinAlg.dpr, а вікно модуля проекту- UnLinAlg.pas.

Давати файлам осмислені імена замість одноманітних Unit1 і Project1, пропонованих Delphi.

Щоб уникнути втрати файлів проекту в аварійних ситуаціях зв'язаних, наприклад, з виключенням блоку живлення, і т.д., рекомендується періодично зберігати проект, використовуючи піктограму  головного меню або опцію Save All в меню File.

4. Налаштування вікон.

Щоб працювати з вікном, необхідно зробити його активним, клацнувши “мишею” в будь-якому місці вікна. Біля активного вікна заголовок стає виділеним, наприклад, на рис. 1.2 активним є вікно Редактора Коду.

Вікна Форми і Редактора Коду модуля в правому верхньому куті мають кнопки управління, які призначені:  – для згортання вікна в піктограму,  – для розвертання вікна на весь екран і повернення до початкового розміру ,  – для закриття вікна.

За допомогою “миші”, захоплюючи одну з кромek вікна або виділений рядок заголовка, відрегулюєте потрібні розміри вікон Форми, Редактора Коду, Інспектора Об'єктів і їх положення на екрані.


5. Зміна заголовка Форми.

Нова Форма має однакові ім'я (Name) і заголовок (Caption) – Form1. Програмістам, що починають, ім'я Форми змінювати не рекомендується, оскільки воно використовується в тексті модуля.

Для зміни заголовка активізуйте вікно Інспектора Об'єктів і на сторінці Properties у властивості Caption замініте заголовок Form1 на *Лабораторна робота №2 – Засоби консольного введення/виведення в Delphi*. Переконайтеся, що одночасно змінився заголовок вікна Форми.


6. Розміщення компонентів на Формі.


Розміщуватимемо компоненти на Формі так, щоб вони відповідали панелі, показаній на рис 1.2.

Для нанесення текстів на Формі використовується компонент **Label**. Виберіть в Палітрі Компонентів на сторінці Standard піктограму  компоненту Label і клацніть на ній “мишею”. Потім в потрібному місці Форми клацніть “мишею” – з'явиться напис Label1. У властивості Caption Інспектора Об'єктів замініть текстове поле Label1 на Введіть значення X:. У властивості Font підберіть шрифт. Аналогічно нанесіть на Форму решту текстів. Клацнувши “мишею” на будь-якому з розміщених компонентів, відрегулюйте його місцеположення на Формі і розмір.

Для введення/виведення консольних додатків в найпростіших випадках використовуються компоненти *Edit* і *Memo*. Компонент Edit застосовується в тих випадках, коли дані представляються одним

рядком. Якщо дані є декількома рядками, то використовується компонент Memo.

Для створення полів введення чисельних значень змінних x , y і z використовуємо компонент Edit. Виберіть в Палітрі Компонентів на сторінці **Standard** піктограму  і розмістіть компонент **Edit** в потрібних місцях Форми так само, як Ви це робили з компонентом Label1.

Для виведення результатів використовуємо компонент **Memo**. Виберіть в Палітрі Компонентів на сторінці Standard піктограму , помістіть компонент Memo на Форму і відкоригуйте його місцеположення і розміри.


7. Написання процедури обробки події створення Форми (*FormCreate*).

Якщо програміст бажає, щоб при появі панелі інтерфейсу на екрані у відповідних полях знаходилися початкові значення даних, він повинен врахувати, що при запуску консольного додатку виникає подія – створення Форми (*OnCreate*). Створимо процедуру обробки цієї події, яка занесе початкові значення змінних x , y , z в поля Edit1, Edit2 і Edit3 відповідно, а в полі Memo1 помістить рядок *Лабораторна робота №2–Засоби консольного введення/виведення в Delphi*.

Для цього двічі клацніть мишею на будь-якому вільному місці Форми. На екрані з'явиться текст модуля UnLinAlg, в якому Delphi автоматично створює заготовку до процедури-обробника події створення Форми:

Procedure TForm1.FormCreate(Sender:TObject). Між операторами *begin* і *end* цієї процедури вставте оператори, які виконують необхідні дії.

8. Написання процедури обробки події натиснення кнопки Button1 (*Button1Click*).

Помістимо на Форму кнопку, натиснення якої приведе до обчислення виразу. Виберіть в Палітрі Компонентів на сторінці **Standart** піктограму  компоненту **Button**. У властивості Caption Інспектора Об'єктів замінити текст на кнопці Button1 на “Обчислити.” У властивості Font підберіть шрифт. Відрегулюйте положення і розмір кнопки. Потім двічі клацніть “мишею” на кнопці, після чого курсор

встановиться в тексті процедури-обробника події натиснення кнопки Button1:

Procedure TForm1.Button1Click(Sender:TObject).

Уважно наберіть оператори цієї процедури, використовуючи текст модуля UnLinAlg.

9. Текст модуля UnLinAlg:

```
Unit UnLinAlg;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs  
StdCtrls;
```

```
type
```

```
TForm1 = class(TForm)  
  Label1: TLabel;  
  Edit1: TEdit;  
  Label2: TLabel;  
  Edit2: TEdit;  
  Label3: TLabel;  
  Edit3: TEdit;  
  Label4: TLabel;  
  Memo1: TMemo;  
  Button1: TButton;  
  procedure FormCreate(Sender: TObject);  
  procedure Button1Click(Sender: TObject);  
private  
  { Private declarations }  
public  
  { Public declarations }  
end;
```

```
var
```

```
Form1: TForm1;
```

```
implementation
```

```
{ $R *.DFM }
```

```
// Процедура обробки події створення Форми:
```

```
procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
```

```
  Edit1.Text:='0.4';      // початкове значення X
```

```

Edit2.Text:='8.75e-2'; // початкове значення Y
Edit3.Text:='-19.63'; // початкове значення Z
Memo1.Clear; // очищення Memo1
//Виведення рядка в Memo1:
Memo1.Lines.Add('Практична робота №1 - Лінійний алгоритм');
end;
// Процедура обробки події натиснення кнопки Button1:
procedure TForm1.Button1Click(Sender: TObject);
var
  x,y,z,a,b,c,u : extended; // оголошення локальних змінних
begin
  x:=StrToFloat(Edit1.Text); // X привласнюється вміст Edit1
  y:=StrToFloat(Edit2.Text); // Біля привласнюється вміст Edit2
  z:=StrToFloat(Edit3.Text); // Z привласнюється вміст Edit3
  // Обчислюється вираз
  a:=sqrt(x)-y*y*y;
  b:=sin(a)/cos(a);
  z:=Exp(5*Ln(b));
  u:=c+exp(y/z)*sin(z*z);
  Memo1.Lines.Add('X = '+Edit1.Text+' Y = '+Edit2.Text+' Z = '+Edit3.Text);
  // контрольний вивід X, Y, Z в Memo1
  //Виведення результату в Memo1:
  Memo1.Lines.Add(' U = '+FloatToStrF(u,ffFixed,8,3));
end;

end.

```

Дані, з якими працюють компоненти **Edit** і **Memo**, мають *тип String*. Тому в процедурі `TForm1.Button1Click` при привласненні вмісту полів **Edit1,Edit2,Edit3** змінним **X,Y,Z** за допомогою функції `StrToFloat` здійснюється перетворення даних типу `String` в дійсні значення з плаваючою точкою типу `Extended`. Якщо необхідно працювати з даними цілого типу, використовується функція `StrToInt`.

При виведення даних в **Memo1** використовується метод **Add** властивості **Lines**, причому для перетворення даних з дійсного значення в рядкове і управління формою представлення результату, що виводиться, використовується функція `FloatToStrF`.


10. Робота з консольним додатком.

Для запуску створеного консольного додатку натисніть піктограму



головного меню або клавішу `F9`. При цьому відбувається компіляція модулів і, якщо немає помилок, компоновка проекту і

створення виконуваного файлу PrLinAlg.exe. На екрані з'являється панель інтерфейсу додатку (рис.1.2).

Клацніть “мишею” на кнопці “Обчислити” і в полі Memo1 з'являється результат. Змініть початкові значення x, Y, z в полях Edit і знову натисніть кнопку ”Обчислити”. Переконайтеся, що в полі Memo1 відображаються нові результати. Завершити роботу консольного додатку можна натисненням кнопки  в правому верхньому кутку панелі інтерфейсу.

У разі нештатного функціонування консольного додатку відновити первинний режим роботи з проектом можна шляхом вибору в меню Run опції ProgramReset або натисніть клавіші Ctrl+F2.

11. Виконання індивідуального завдання.

По вказівці викладача виберіть своє індивідуальне завдання. Уточніть умову завдання, кількість і типи початкових даних. Відповідно до цього оформіть дизайн панелі інтерфейсу проектного консольного додатку, встановіть необхідну кількість полів Edit, тексти заголовків на Формі, розміри шрифтів, а також типи змінних і функції перетворення при введенні і виведенні результатів.

Індивідуальні завдання:

$$1. t = \frac{2 \cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y} \left(1 + \frac{z^2}{3 - z^2/5}\right).$$

$$\text{При } x=14.26, y=-1.22, z=3.5 \times 10^{-2} \quad t=0.564849.$$

$$2. u = \frac{\sqrt[3]{8 + |x - y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|} (tg^2 z + 1)^x.$$

$$\text{При } x=-4.5, y=0.75 \times 10^{-4}, z=0.845 \times 10^2 \times 10^2 \quad u=-55.6848.$$

$$3. v = \frac{1 + \sin^2(x + y)}{\left|x - \frac{2y}{1 + x^2 y^2}\right|} x^{|y|} + \cos^2\left(\arctg \frac{1}{z}\right).$$

$$\text{При } x=3.74 \times 10^{-2}, y=-0.825, z=0.16 \times 10^2 \times 10^2 \quad v=1.0553.$$

$$4. w = |\cos x - \cos y|^{(1+2 \sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right).$$

При $x=0. \times 10^4$, $y=-0.875$, $z=-0.475 \times 10^{-3} \times 10^{-3}$ $w=1.9873$.

$$5. \alpha = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \frac{y}{2}\right) + \sin^2 \arctg(z).$$

При $x=-15.246$, $y=4.642 \times 10^{-2}$, $z=20.001 \times 10^2 \times 10^2$ $=-182.036$.

Вимоги до оформлення звіту:

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторної роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Для чого призначено вікно інспектора об'єктів?
2. Що таке вікно форми?
3. Для чого призначено редактор коду?
4. Що забезпечує палітра компонентів?
5. Які знаєте обробники подій, які приймає Delphi – їхні значення?
6. Як настроювати вікна?
7. За допомогою якого компонента заносите текст на форму?
8. Як працювати з консольним додатком?
9. Що таке консольний додаток?

Практична робота №3.

Тема: Створення програми «Обмін валюти».

Мета: Навчитись розв'язувати лінійні програмні структури. Вивчення об'єктів поля редагування та кнопки.

Теоретичні відомості.

Об'єкт поле редагування, інша назва – поля введення/виведення (Edit), має властивість Text, яка дає змогу користувачеві задати текстове значення для даного поля. Якщо треба ввести число, то число й уводять, але програмою воно розглядатиметься як текстове дане типу string. Тому в підпрограмах користувач має застосувати процедуру Val (чи іншу) для переведення текстових даних з зображеннями чисел у відповідні числа типу integer чи real тощо.

Виводять результати перетворення даних також у поля типу Edit. Якщо треба вивести число, то спочатку його слід перевести у дане типу string за допомогою стандартної для мови Паскаль процедури Str.

Об'єкт кнопка (Button) призначений для керування процесами, що відбуваються на формі. Він має такі властивості: заголовок, шрифт, розміри, колір, координати розташування на формі тощо.

Для об'єктів поле редагування та кнопка можна задати такі стандартні події: OnClick, OnEnter, OnExit і поставити їм у відповідність реакції (процедури користувача). Наприклад, процедура-реакція на подію OnExit для кнопки може складатися з однієї команди close, яка призначена для закриття вікна. Подія OnDblClick (подвійне клацання мишею) для об'єкту кнопка не визначена.

Завдання :

Створимо форму з назвою «Обмін валюти» для розв'язування такої задачі: перевести в гривні задану грошову суму у євро згідно з курсом валют.

Для розв'язування задачі на формі потрібно розташувати поля редагування для введення значень курсу, кількості доларів та виведення суми в гривнях,, а також кнопки для виконання обчислень та закінчення роботи програми.

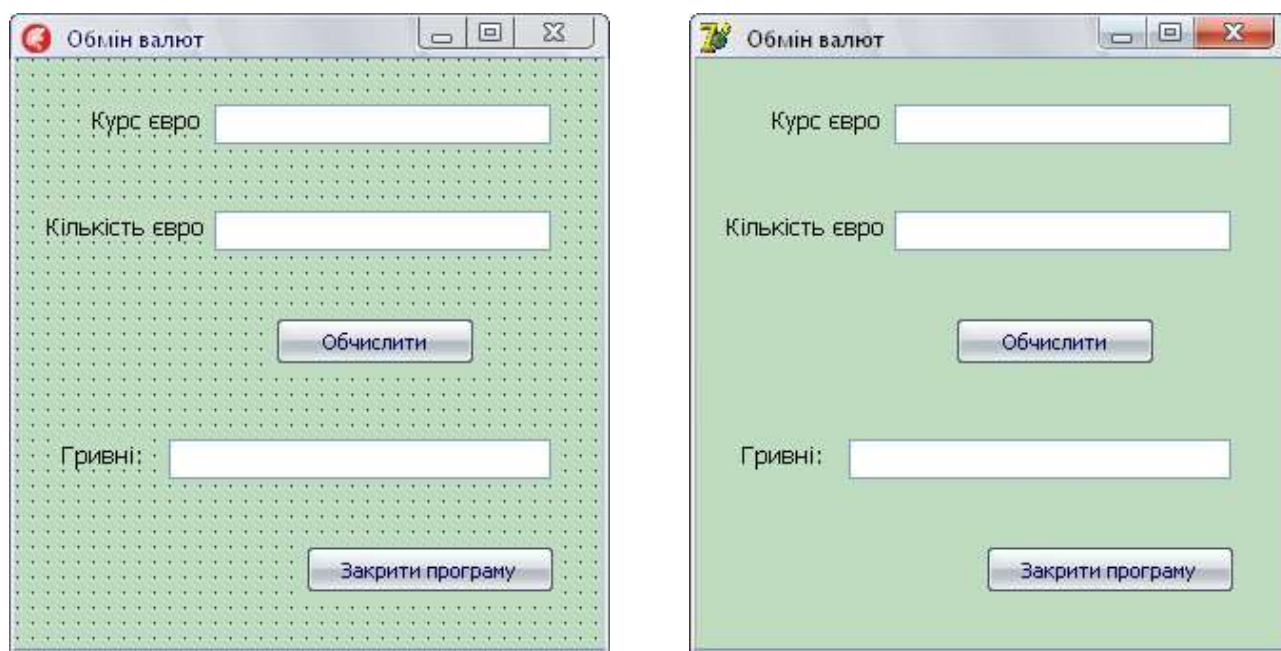
Хід роботи.

Завантажте систему візуального програмування Delphi.

Задайте заголовок, розмір та колір форми. Для цього у вікні Object Inspector уведіть значення наступних властивостей: Caption (Заголовок форми) – Обмін валюти, колір та розмір форми задайте на свій вибір.

3. Збережіть створену програму у власній папці. Для цього виберіть команду Save All,(Зберегти все) головного меню File. Задайте імена програмного модуля та проекту.

Запустіть програму на виконання. Для цього натисніть на клавішу F9 або на кнопку Run. Розгляньте і закрийте отримане вікно програми. Вставте у, форму текстові поля Label1 – Label3 як показано на рис. 1 і надайте їм відповідні властивості. Для цього клацніть, мишею на піктограмі Label на закладці Standard з палітри компонентів, а потім – у тому місці на формі, де має знаходитись текстове поле. Активізуйте це поле і у вікні Object Inspector задайте властивість Caption: для Label1 – "Курс євро", для Label2 – "Кількість євро", для Label3 – "Гривні" (рис. 2).



5. Вставте у форму три поля редагування (об'єкти типу Edit) як показано на рис.1 і надайте їм відповідні властивості. Для цього клікніть мишею на піктограмі Edit на закладці Standard, а потім – на формі, де має знаходитись поле редагування. Очистити поля редагування Edit3 (виперти значення властивості Text цих об'єктів). Для поля Edit3 заблокуйте можливість уведення даних, оскільки це поле міститиме результат. Для цього надати значення False властивості **Enabled** (доступність).

6. Вставте у форму дві кнопки як показано на рис.1 і надайте їм відповідні назви (рис.2 властивості Caption для кнопки Button1 – «Обчислити», а для кнопки Button2 – «Закрити програму» або «Вийти».

7. Збережіть усе (див. пункт 3).

8. Запустіть програму на виконання. Закрийте вікно програми.

9. **Запрограмуйте кнопку «Обчислити».** Для цього клацніть двічі лівою клавішею миші на кнопці “Обчислити” і введіть у заготовку процедури, яка вже є на екрані:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
<вводити потрібно сюди>  
end;  
end.
```

наступний текст програми:

```
Val(edit1 .text,course,code); {Отримаємо значення курсу долара}  
Val(edit2.text,sum,code);    {Отримаємо значення кількості доларів}  
sum:=sum*course;  
Str(sum: 10:2,grn);          {Суму (число) перетворюємо на текстовий формат}  
Edit3.Text:=grn;             {Формуємо значення властивості Text об'єкта EditS}
```

Перед словом begin вставте розділ опису змінних:

```
var course,sum:real; code:integer; grn:string[10];
```

11. Перейдіть на форму і запрограмуйте кнопку «Закрити програму» так:

```
procedure TForm1.Button2Click(Sender: TObject);  
Begin  
Close  
End;
```

12. Збережіть створену форму (див. пункт 3).

13. Запустіть програму на виконання.

14. Створіть exe-файл програми. Виконайте команду головного меню Project → Build All (Сконструювати Все).

15. Продемонструйте створену форму викладачу.

Самостійне завдання:

Написати програму переводу євро у гривні та гривні у євро.

Вимоги до оформлення звіту:

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторної роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Як вставити у форму поле редагування?
2. Які властивості поля редагування ви знаєте?.
3. Чим відрізняється поле редагування від текстового поля?
4. Як вставити кнопку на форму?
5. Як запрограмувати кнопку?
6. Яку команду закриття вікна ви знаєте?

Практична робота №4.

Тема: Створення програми обміну валюти у двох напрямках .

Мета: Навчитись розв'язувати програми з розгалуженням обчислювальним процесом.. Вивчення об'єкту *RadioButton* та функції *MessageBox*.

Теоретичні відомості.

1. Вікно повідомлень.

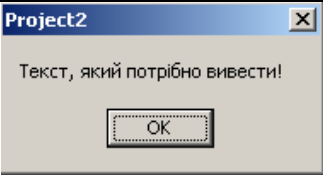
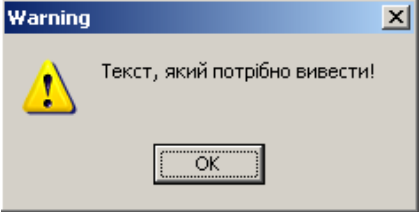
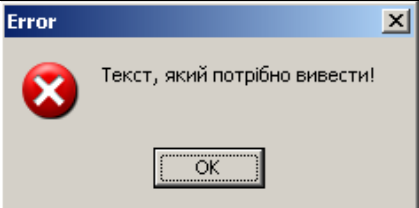
Шаблон:

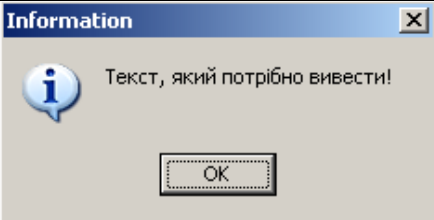
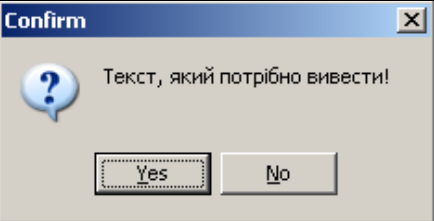
`MessageBox(const Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint)`

розділ Help) стрічка; тип вікна; кнопки у вікні;


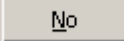

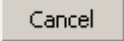


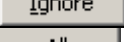
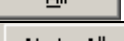
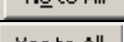

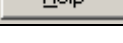
Опис: MessageBox потрібне для діалогу з користувачем.

- 1) Msg – записуємо стрічку, яку буде виводити вікно.
- 2) DlgType показує, який буде тип вікна та що буде у ньому міститися. Можна вибрати наступні варіанти TMsgDlgType:

mtCustom	Заголовок вікна буде співпадати з назвою проекту	
mtWarning	У вікні з'явиться жовтий трикутник з знаком оклику	
mtError	У вікні з'явиться знак Stop	

mtInformation	У вікні з'являється голубе „i”, що означає, що вікно є просто інформаційне	
mtConfirmation	Порібно вибрати „Так” чи „Ні”. Вікно підтвердження.	

3) Buttons вказує на те, які кнопки мають з'явитися у вікні. Можна вибрати наступні варіанти TmsgDlgButtons:

mbYes	Кнопка „Yes”	
mbNo	Кнопка „No”	
mbOK	Кнопка „OK”	
mbCancel	Кнопка „Cancel”	
mbAbort	Кнопка „Abort”	
mbRetry	Кнопка „Retry”	
mbIgnore	Кнопка „Ignore”	
mbAll	Кнопка „All”	
mbNoToAll	Кнопка „NoToAll”	
mbYesToAll	Кнопка „YesToAll”	
mbHelp	Кнопка „Help”	

Крім цього можна використовувати такі TmsgDlgButtons:

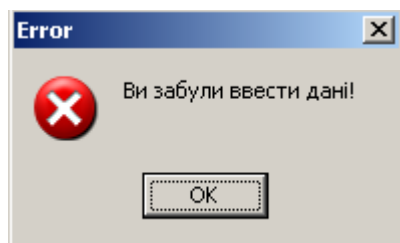
mbYesNoCancel	Поєднання кнопок „Yes”, „No” та „Cancel”
mbYesNoAllCancel	Поєднання кнопок „Yes”, „No”, „All” та „Cancel”
mbOKCancel	Поєднання кнопок „OK” та „Cancel”
mbAbortRetryIgnore	Поєднання кнопок „Abort”, „Retry” та „Ignore”
mbAbortIgnore	Поєднання кнопок „Abort” та „Ignore”

4) HelpCtx визначає значення ID розділу Help, коли користувач натискає на кнопку «Help» або на клавішу F1.

Приклад запису:

MessageDlg('Ви забули ввести дані!', mtError, [mbOk], 0)

Вигляд того, що ми отримуємо:



2. Перетворення числа у стрічку та навпаки.

1) Перетворення числа у стрічку

цілого:

функція `IntToStr(ціле число)` – повертає стрічку, яка є результатом перетворення цілого числа;

дробового:

функція `FloatToStr(дробове число)` – повертає стрічку, яка є результатом перетворення дробового числа;

будь-якого:

`Str(N,S)` – перетворення числа `N` у стрічку `S`.

2) Перетворення стрічки у число:

цілого:

функція `StrToInt(стрічка)` – повертає ціле число, яке є результатом перетворення стрічки;

дробового:


функція `StrToFloat(стрічка)` – повертає дробове число, яке є результатом перетворення стрічки;

будь-якого:

`Val(S,V,C)` – перетворення стрічки `S` у число `V`, а `C` – це код помилки (номер символу, який неможливо перетворити). Якщо перетворення здійснено успішно, то `C=0`.

3. Перемикач.

Перемикачі представляють собою набір взаємовиключаючих виборів. Тобто, якщо можливо вибрати лише один з перемикачів.

Вибраний перемикач представляє собою заповнений круг .

Перемикач, який є пасивним на даний момент має наступний вигляд:

 `RadioButton2`.

Завдання:

Створимо форму з назвою «Обмін валюти» для розв'язування такої задачі: згідно курсу валют перевести задану грошову суму в гривнях - у євро та навпаки.

Для розв'язування задачі на формі потрібно розташувати поля редагування для введення значень курсу, кількості грошей та виведення суми, перемикачі для вибору напрямку переведення грошей, а також кнопки для виконання обчислень та закінчення роботи програми.

Хід роботи.

1. Завантажте систему візуального програмування Delphi.
2. Задайте заголовок, розмір та колір форми. Для цього у вікні Object Inspector уведіть значення наступних властивостей: Caption (Заголовок форми) – ‘Обмін валюти’, колір та розмір форми задайте на свій вибір.
3. Збережіть створену програму у власній папці. Для цього виберіть команду Save All (Зберегти все) головного меню File. Задайте імена програмного модуля та проекту.
4. Вставте у форму текстові поля Label1-Label3 як показано на рис.1 і надайте їм відповідні властивості. Для цього клікніть мишею на піктограмі Label на закладці Standard з палітри компонентів, а потім – у тому місці на формі, де має знаходитись текстове поле. Активізуйте це поле і у вікні Object Inspector задайте властивість Caption: для Label1 – “Курс”, для Label2 – “Кількість грошей”, для Label3 – “Валюта, яку міняєте” (рис. 4.2).
5. **RadioButton1 → Checked → True**

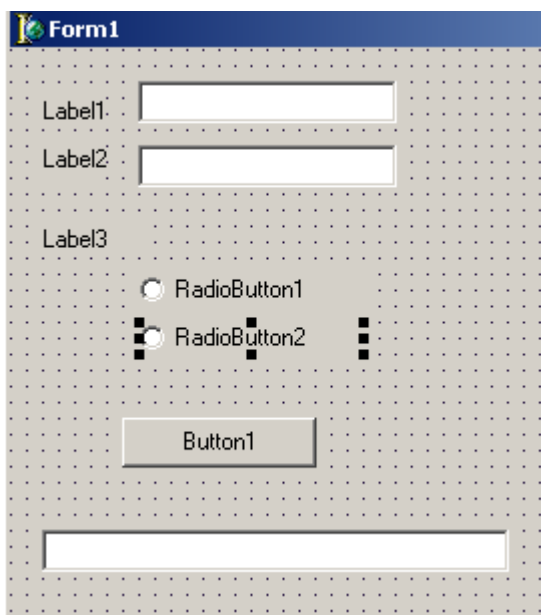


рис. 4.1

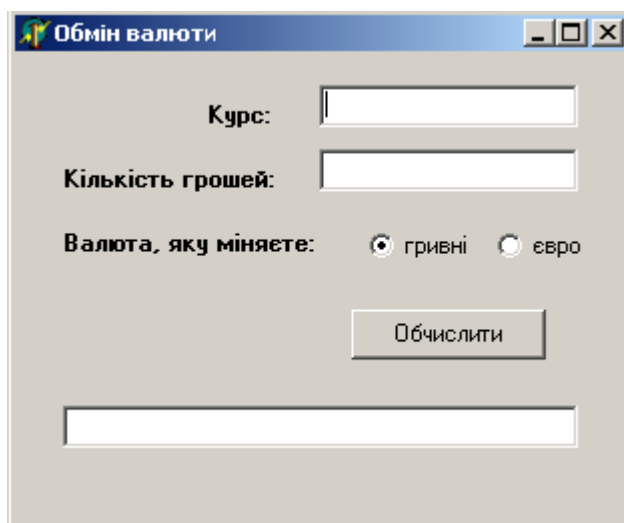


рис. 4.2

6. Вставте у форму три поля редагування (об'єкти типу Edit) як показано на рис.1 і надайте їм відповідні властивості. Для цього клікніть мишею на піктограмі Edit на закладці Standard, а потім – на

формі, де має знаходитись поле редагування. Очистити поля редагування Edit1-3 (випередити значення властивості Text цих об'єктів). Для поля Edit3 заблокуйте можливість введення даних, оскільки це поле міститиме результат. Для цього надайте значення False властивості Enabled (доступність).

7. Вставте у форму дві кнопки як показано на рис.1 і надайте їм відповідні назви (рис. 4.2 властивості Caption для кнопки Button1 – «Обчислити», а для кнопки Button2 – «Закрити програму» або «Вийти».

8. Збережіть усе (див. пункт 3).

9. Запустіть програму на виконання. Закрийте вікно програми.

10. **Запрограмуйте кнопку «Обчислити».** Для цього клацніть двічі лівою клавішею миші на кнопці «Обчислити» і введіть у заготовку процедури, яка вже є на екрані:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  <вводити потрібно сюди>
end;
end.
наступний текст програми:
Val(edit1.text,course,code); {Отримаємо значення курсу долара}
Val(edit2.text,sum,code);   {Отримаємо значення кількості доларів}
sum:=sum*course;
Str(sum:10:2,grn);          {Суму (число) перетворюємо на текстовий
формат}
Edit3.Text:=grn;           {Формуємо значення властивості Text об'єкта
EditS}
```

Перед словом begin вставте розділ опису змінних:

```
var course,sum:real; code:integer; grn:string[10];
```

11. Потрібно описати, що відбудеться, коли або поле Edit1, або поле Edit2 залишаться порожніми. Для цього напишемо наступне:

- якщо поле Edit1 порожнє, тоді:

```
if Edit1.Text="" then    {" –лапки повинні бути одинарні}
begin
  MessageDlg('Потрібно ввести курс!',mtError,[mbOk], 0);
  Edit1.SetFocus        {після натискання на кнопку ОК у вікні повідомлень
курсор переходить у поле Edit1}
end;
```

якщо поле Edit2 порожнє, тоді:

```
if Edit2.Text=""then
```

```
begin
```

```
    MessageDlg('Потрібно ввести кількість грошей!',mtError,[mbOk], 0);
```

```
    Edit2.SetFocus    {після натискання на кнопку ОК у вікні повідомлень  
курсор переходить у поле Edit2}
```

```
end;
```

12. Перемикачі потрібні для того, щоб визначити напрямок обчислень. Коли поля Edit1 та Edit2 не порожні, тоді, в залежності від активного перемикача, проводяться обчислення:

```
If not(Edit1.Text="") and not(Edit2.Text=") then
```

```
begin
```

```
If Radiobutton1.Checked then
```

```
    Edit3.Text:=FloatToStr(StrToFloat(Edit2.Text)/StrToFloat(Edit1.Text))+ 'євро';
```

```
    if Radiobutton2.Checked then
```

```
        Edit3.Text:=FloatToStr(StrToFloat(Edit1.Text)*StrToFloat(Edit2.Text))+ 'гривень';
```

```
end;
```

13.Збережіть створену форму (див. пункт 3).

14.Запустіть програму на виконання.

15.Створіть exe-файл програми. Виконайте команду головного меню Project → Build All (Сконструювати Все).

16.Продемонструйте створену форму викладачу.

Вимоги до оформлення звіту:

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.
5. Контрольні запитання:
6. Як вставити у форму перемикач?
7. Який шаблон запису вікна повідомлень (українською мовою)?
8. Які бувають типи вікна повідомлень?
9. Які бувають кнопки у вікні повідомлень?
- 10.Як перетворити число у стрічку?
- 11.Як перетворити стрічку у число?
- 12.Що таке перемикач?

Практична робота №5.

Тема: **Програмування основних циклічних алгоритмів**

Мета: освоїти найпростіші засоби відладки модулів проекту і створити додаток, який використовує циклічний алгоритм.

Теоретичні відомості:

1. Відкладка модулів проекту.

Відкладка є процесом виявлення, локалізації і усунення помилок в проекті. Вона займає значну частину робочого часу програміста, нерідко більшу, ніж розробка проекту.

Практично будь-який нетривіальний проект перед початком відладки містить хоча б одну синтаксичну або логічну помилку.

2. Відладка синтаксичних помилок.

Синтаксичні помилки полягають в порушенні формальних правил використання операторів. Ці помилки з'являються в результаті недостатнього знання розробником мови програмування і неуважності при наборі операторів на екрані дисплея.

Пошук синтаксичних помилок в модулях проекту здійснюється компілятором. Щоб дати програмісту якомога більше інформацію про помилки, допущені в модулі, компілятор наголошує на помилках і продовжує роботу до тих пір, поки не будуть оброблені всі оператори модуля. Слід мати у вигляді, що:

- 1) компілятор розпізнає *не всі помилки*;
- 2) деякі помилки можуть спричинити за собою те, що правильні оператори сприйматимуться компілятором як помилкові, і навпаки – помилкові оператори компілятор прийме як вірні;
- 3) помилка в одному місці модуля може спричинити за собою серію діагностичних повідомлень компілятора в інших місцях модуля;
- 4) через деякі помилки компіляція модуля може взагалі припинитися і перевірка подальших операторів не проводиться.

Інформація про всі помилки, знайдені в модулі, виводиться в спеціальне вікно, яке з'являється в нижній частині екрану. Кожний рядок цього вікна містить ім'я файлу, номер рядка, в якому знайдена помилка і характер помилки. Якщо двічі клацнути “мишею” на рядку з описом помилки, курсор встановиться в тому рядку модуля, де знайдена помилка. Слід виправляти помилки послідовно, зверху вниз і після виправлення кожної помилки компілювати програму наново. З метою скорочення часу компіляції рекомендується здійснювати

перевірку наявності помилок в режимах Syntax Check і Compile меню Project. Для отримання більш повної інформації про характер помилки можна звернутися до Help натисненням клавіші F1.

Відкладка синтаксису вважається завершеною, коли після чергової компіляції в режимі Build All меню Project відсутні діагностичні повідомлення про помилки.

3. Відкладка логічних помилок.

Логічні помилки умовно можна розділити на помилки алгоритму і семантичні помилки. Причинами таких помилок можуть бути невідповідність алгоритму поставленій задачі, неправильне розуміння програмістом значення (семантики) операторів мови програмування, порушення допустимих меж і правил представлення даних, неуважність при технічній підготовці проекту до обробки на комп'ютері.

Для виявлення помилок служать *тести*. Тест – це такий набір початкових даних, який дає результат, що не викликає сумнівів. Проміжні і кінцеві результати тесту використовуються для контролю правильності виконання додатку.

Складання тестів – непроста задача. Тести повинні бути з одного боку, достатньо простими, щоб результат легко перевірявся, з другого боку – достатньо складними, щоб комплексно перевірити алгоритм.

Тести складаються по схемі алгоритму *до програмування*, оскільки складання тестів допомагає виявити багато помилок в алгоритмізації.

Кількість тестів і їх складність залежать від алгоритму. Комплекс тестів повинен бути таким, щоб всі гілки схеми алгоритму були пройдені, принаймні, по одному разу. Неспівпадань результатів, видаваних додатком з результатами тестів – ознака наявності помилок. Ці помилки виявляються в тому, що результат розрахунку виявляється невірним або відбувається переповнювання, розподіл на 0 і ін.

Для локалізації місця помилки рекомендується поступати таким чином. У вікні Редактора Коду встановіть курсор в рядку перед підозрілою ділянкою і натискуйте клавішу F4 (виконати до курсора). Виконання додатку буде зупинено на тому рядку модуля, в якому був встановлений курсор. Поточне значення будь-якої змінної можна побачити, якщо накрити курсором ідентифікатор змінної на 1-2 сек. Натискуючи клавішу F8 (покрокове виконання), можна відрядковий виконувати програму, контролюючи вміст змінних і правильність обчислень.

4. Приклад створення додатку.

Завдання:

Створити Windows-додаток, який виводить таблицю значень функції : $Y(x) = (1 - \frac{x^2}{2}) \cos x - \frac{x}{2} \sin x$ і її розкладання в ряд у вигляді суми

$$S(x) = \sum_{n=0}^n (-1)^n \frac{2n^2 + 1}{(2n)!} x^{2n}$$
 для значень x від x_n до x_k з кроком

$$h=(x_k - x_n)/10.$$

У панелі інтерфейсу передбачити можливість управління висновком початкових даних і обчислень. Один з можливих варіантів панелі інтерфейсу створюваного додатку показаний на рис.5.1.

5. Розміщення компонентів на Формі.

Замість компоненту Edit використовуємо компонент SpinEdit, який забезпечує відображення і редагування цілого числа з можливістю його зміни за допомогою подвійної кнопки.

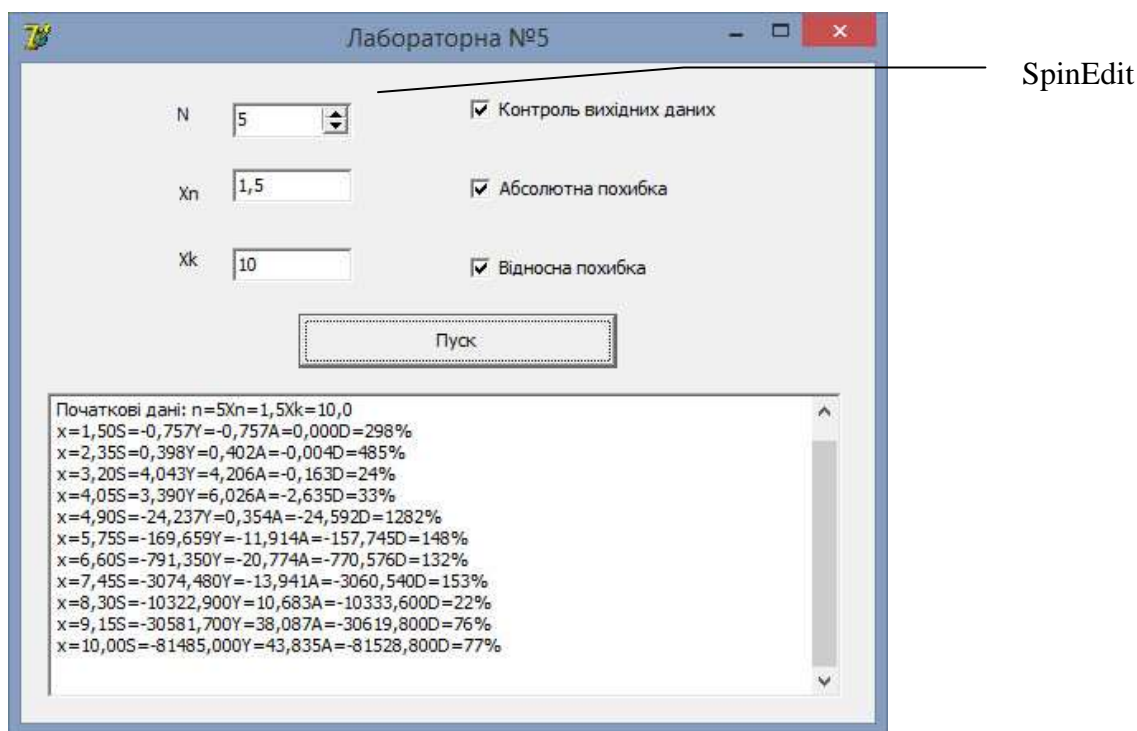


Рис. 5.1

Компонент SpinEdit знаходиться на сторінці Samples Палітри Компонентів. В тих випадках, коли об'єм інформації, що виводиться, перевищує розмір поля компоненту Мето, доцільно забезпечити його

лінійками прокрутки. У властивості ScrollBars компоненту Memo1 встановимо значення ssVertical – з'явиться вертикальна лінійка прокрутки. Привласнимо модулю ім'я UnCiklAlg.

6. Текст модуля UnCiklAlg:

```
Unit UnCiklAlg;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs
  StdCtrls, ExtCtrls, Spin;
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    SpinEdit1: TSpinEdit;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
  SpinEdit1.text:='3'; // початкове значення N
  Edit1.text:='0.1'; // початкове значення Xn
  Edit2.text:='2.0'; // початкове значення Xk
  Memo1.Clear;
  Memo1.Lines.Add('Практична робота №5 - Циклічний алгоритм');
end;
```

```

procedure TForm1.Button1Click(Sender: TObject);
var xn,xk,x,h,c,s,y,al,del:extended;
    n,k:integer;
begin
n:=StrToInt(SpinEdit1.Text);
xn:=StrToFloat(Edit1.Text);
xk:=StrToFloat(Edit2.Text);
if CheckBox1.Checked then
    Memo1.Lines.Add('Исходные дані: n='+IntToStr(n)+
        ' Xn='+FloatToStrF(xn,ffFixed,6,1)+
        ' Xk='+FloatToStrF(xk,ffFixed,6,1));
h:=(xk-xn)*0.1;    // крок h
x:=xn;
repeat            // цикл по x
c:=-x*x*0.5;
S:=1;
for k:=1 to n do
begin
s:=s+c*(2*k*k+1);
z:=-c*x*x/((2*k+1)*(2*k+2));
end;
Y=(1-x*x*0.5)*cos(x)-0.5*x*sin(x);
if CheckBox2.Checked then
if CheckBox3.Checked then
begin
al:=s-y;                // абсолютна погрiшнiсть
del:=abs((s-y)/y)*100; // вiдносна погрiшнiсть
Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
    ' S='+ FloatToStrF(s,ffFixed,6,3)+
    ' Y='+ FloatToStrF(y,ffFixed,6,3)+
    ' A='+ FloatToStrF(al,ffFixed,6,3)+
    ' D='+ FloatToStrF(del,ffFixed,6,0)+'%');
end
else
begin
al:=s-y;
Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
    ' S='+ FloatToStrF(s,ffFixed,6,3)+
    ' Y='+ FloatToStrF(y,ffFixed,6,3)+
    ' A='+ FloatToStrF(al,ffFixed,6,3));
end
else
if CheckBox3.Checked then
begin
del:=abs((s-y)/y)*100;

```

```

Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
  ' S='+ FloatToStrF(s,ffFixed,6,3)+
  ' Y='+ FloatToStrF(y,ffFixed,6,3)+
  ' D='+ FloatToStrF(del,ffFixed,6,0)+'%');
end
else
Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,6,2)+
  ' S='+ FloatToStrF(s,ffFixed,6,3)+
  ' Y='+ FloatToStrF(y,ffFixed,6,3));
x:=x+h;
until x>xk;
end;
end.

```

7. Виконання індивідуального завдання:

По вказівці викладача виберіть своє індивідуальне завдання. Створіть додаток і протестуйте його роботу.

Індивідуальні завдання:

У завданнях з №1 по №15 необхідно вивести на екран таблицю значень функції $Y(x)$ і її розкладання в ряд $S(x)$ для значень x від x_n до x_k з кроком $h = (x_k - x_n) / 10$. Близькість значень $S(x)$ і $Y(x)$ у всьому діапазоні значень x вказує на правильність обчислення $S(x)$ і $Y(x)$.

№	x_n	x_k	$S(x)$	n	$Y(x)$
1	0.1	1	$x - \frac{x^3}{3!} + \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$	16	$\sin x$
2	0.1	1	$1 + \frac{x^2}{2!} + \dots + \frac{x^{2n}}{(2n)!}$	10	$\frac{e^x + e^{-x}}{2}$
3	0.1	1	$1 + \frac{\cos \frac{\pi}{4}}{1!} x + \dots + \frac{\cos n \frac{\pi}{4}}{n!} x^n$	12	$e^{x \cos \frac{\pi}{4}} \cos(x \sin \frac{\pi}{4})$
4	0.1	1	$1 - \frac{x^2}{2!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!}$	8	$\cos x$
5	0.1	1	$1 - \frac{x^2}{1!} + \frac{x^4}{2!} - \frac{x^6}{3!} + \dots + (-1)^n \frac{x^{2n}}{n!}$	14	e^{-x^2}
6	0.1	1	$x + \frac{x^3}{3!} + \dots + \frac{x^{2n+1}}{(2n+1)!}$	8	$\frac{e^x - e^{-x}}{2}$

7	0.1	1	$\frac{x^3}{3} - \frac{x^5}{15} + \dots + (-1)^{n+1} \frac{x^{2n+1}}{4n^2 - 1}$	12	$\frac{1+x^2}{2} \operatorname{arctg} x - \frac{x}{2}$
8	0.1	1	$1 + \frac{2x}{1!} + \dots + \frac{(2x)^n}{n!}$	10	e^{2x}
9	0.1	1	$1 + 2\frac{x}{2} + \dots + \frac{n^2 + 1}{n!} \left(\frac{x}{2}\right)^n$	14	$\left(\frac{x^2}{4} + \frac{x}{2} + 1\right) e^{\frac{x}{2}}$
10	0.1	0.5	$x - \frac{x^3}{3} + \dots + (-1)^n \frac{x^{2n+1}}{2n+1}$	15	$\operatorname{arctg} x$
11	0.1	0.8	$\frac{x^2}{2} - \frac{x^4}{12} + \dots + (-1)^{n+1} \frac{x^{2n}}{2n(2n-1)}$	10	$x \operatorname{arctg} x - \ln \sqrt{1+x^2}$
12	0.1	1	$-\frac{(2x)^2}{2} + \frac{(2x)^4}{24} - \dots + (-1)^n \frac{(2x)^{2n}}{(2n)!}$	8	$2(\cos^2 x - 1)$

Вимоги до оформлення звіту :

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке відладка?
2. В чому полягають синтаксичні помилки відладки?
3. Що таке логічні помилки ?
4. Яким чином складаються тести?
5. Що таке текст ?
6. Що робить клавіша F8?
7. Що являє собою компонент SpinEdit, його дії?

Практична робота №6.

Тема: Розв'язування задач з циклічним обчислювальним процесом табулювання функції та з заданим числом повторень.

Мета: оволодіти практичними навиками розробки та програмування обчислювального процесу циклічної структури табулювання функції, отримання навиків по створенню головного меню.


Приклад виконання лабораторної роботи.

Завдання:

Скласти блок-схему та програму табулювання функції: $y = \frac{a+b}{e^x + \cos x}$,

для $a=1,3$; $b=1,29$; $x \in [1,2]$; $\Delta x=0,1$

1. Завантажте систему візуального програмування Delphi.
2. Задайте заголовок, розмір та колір форми. Для цього у вікні Object Inspector уведіть значення наступних властивостей: Caption (Заголовок форми) – ‘Практична робота’, колір та розмір форми задайте на свій вибір.
3. Збережіть створену програму у власній папці. Для цього виберіть команду *Save All* (Зберегти все) головного меню File. Задайте імена програмного модуля та проекту.
4. Вставте у форму текстові поля Label1-Label5 як показано на рис.1 і надайте їм відповідні властивості. Для цього клікніть мишею на піктограмі Label на закладці Standard з палітри компонентів, а потім – у тому місці на формі, де має знаходитись текстове поле. Активізуйте це поле і у вікні Object Inspector задайте властивість Caption: для Label1 – ‘Початкове значення x :’ і т.д. відповідно (рис. 2).
5. Вставте у форму поля редагування (об'єкти типу Edit) як показано на рис.1 і надайте їм відповідні властивості. Для цього клікніть мишею на піктограмі Edit на закладці Standard, а потім – на формі, де має знаходитись поле редагування. Очистити поля редагування Edit1-5 (втерти значення властивості Text цих об'єктів).
6. Розмістіть компонент Memo відповідно рис.1. Задайте для цього поля вертикальну та горизонтальну прокрутки. Для цього увімкніть значення ssBoth для властивості *ScrollBars*. Для властивості Lines витріть значення Memo1, а введіть текст відповідно рис.2.

Вставте у форму головне меню (об'єкт типу MainMenu ). Розташуйте піктограму у довільному місці форми.

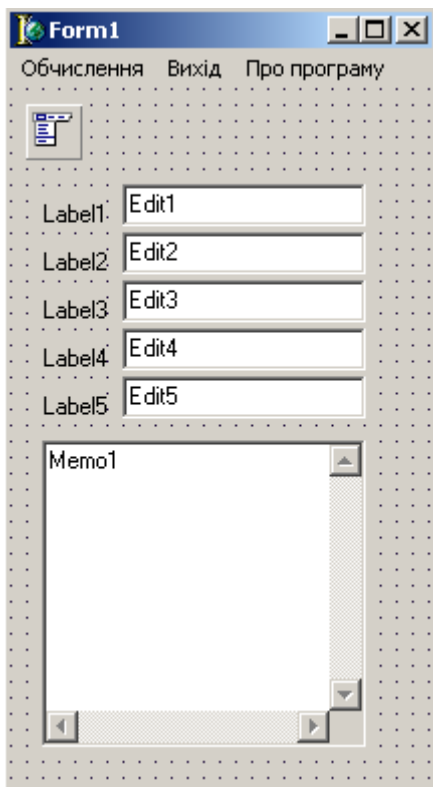


Рис. 6.1

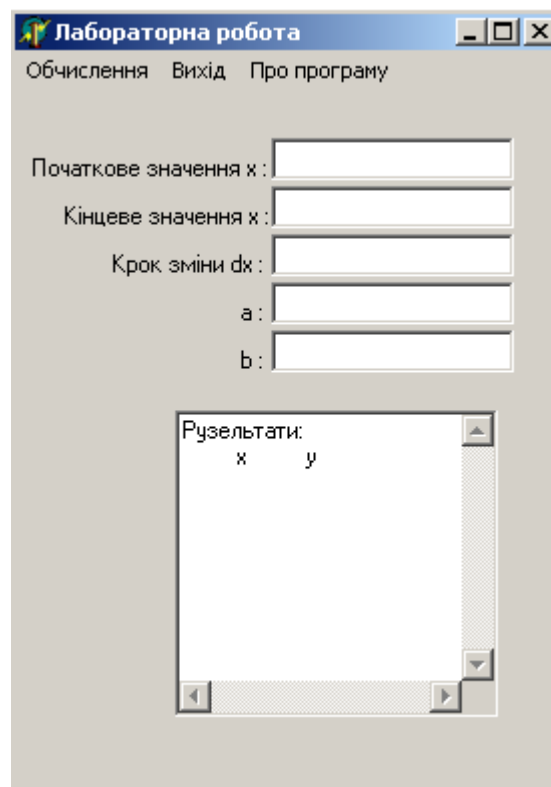


Рис. 6.2

7. Уведіть назви команд головного меню. Для цього двічі клікнути лівою клавiшею миші на вставленому об'єкті MainMenu1. У вікні, що з'явиться (рис. 6.3) виберіть рамку для введення тексту і введіть *Обчислити* у властивості Caption. Підпунктами введіть *Протабулювати функцію*, *очистити поле виведення* (рис. 6.4). Перейдіть у сусідню рамку і запишіть *Вихід* та підпункт *Закрити програму* (рис. 6.5). Так само *Про автора – Автор* (рис. 6.6). Закрити вікно створення команд головного меню Form1.MainMenu1.

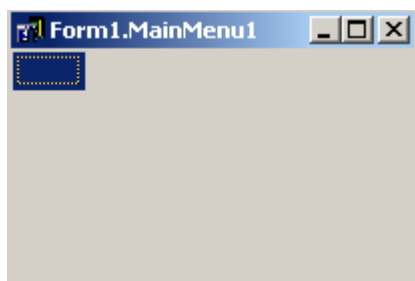


Рис. 6.3

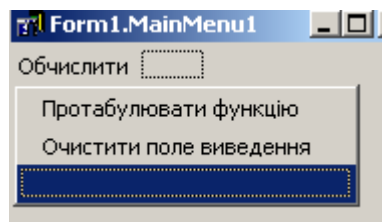


Рис. 6.4

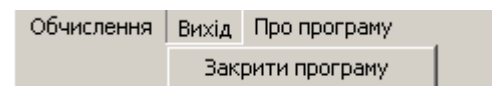


Рис. 5

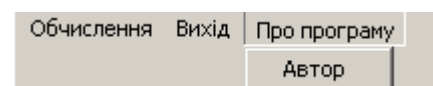


Рис. 6.5

8. Збережіть роботу на диску.

9. Запрограмуйте команду „Протабулювати функцію” головного меню. Для цього клікніть лівою клавiшею миші на команді „Протабулювати функцію” і введіть текст процедури:

```

procedure TForm1.N2Click(Sender: TObject);
var
  x,x0,xk,dx,a,b,y:real;
  s1,s2:string;
begin
x0:=StrToFloat(Edit1.Text);
xk:=StrToFloat(Edit2.Text);
dx:=StrToFloat(Edit3.Text);
a:=StrToFloat(Edit4.Text);
b:=StrToFloat(Edit5.Text);
x:=x0;
repeat
y:=(a+b)/(exp(x)+cos(x));
str(x:10:2,s1);
str(y:10:2,s2);
Memo1.Lines.Add(s1+s2);
x:=x+dx
until x>xk
end;

```

10. Поверніться на форму і запрограмуйте команду „*Очистити поле виведення*” головного меню:

```

procedure TForm1.N3Click(Sender: TObject);
begin
Memo1.Clear
end;

```

11. Поверніться на форму і запрограмуйте команду „*Закрити програму*” головного меню:

```

procedure TForm1.N6Click(Sender: TObject);
begin
Close
end;

```

12. Поверніться на форму і запрограмуйте команду „*Автор*” головного меню (рис.8):

```

procedure TForm1.N7Click(Sender: TObject);
begin
MessageDlg('Програму виконав студент групи ЕК-1/1 Іванов І.В.',mtInformation,[mbOk],0);
end;

```

14. Збережіть програму.

15. Запустіть програму на виконання. Загальний вигляд програми після виконання рис. 6.6.

16. Створіть ехе-файл.

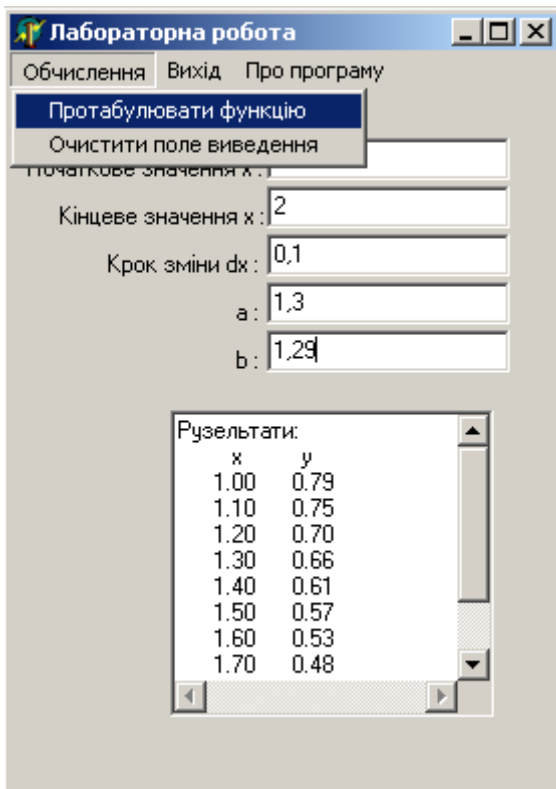


Рис. 6.6

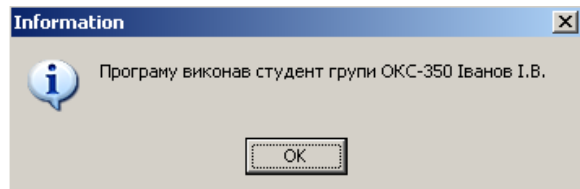


Рис. 6.7

Вимоги до оформлення звіту :

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке компонент Метод?
2. Що таке об'єкт типу MeinMenu, його властивості?
3. Як запрограмувати команду "Протабулювати функцію" головного меню?
4. Як запрограмувати команду "Очистити поле виведення" головного меню?
5. Як запрограмувати команду "Закрити програму"?
6. Як запрограмувати команду "Автор"?
7. Як зберегти програму?

ПРАКТИЧНА РОБОТА №7.

Тема: Розробка алгоритмів і програм з використанням множин.

Мета: оволодіти практичними навиками роботи з множинами.

Хід роботи.

Завдання:

Введена стрічка у поле Edit. Написати програму, яка окремо виводить, які літери та цифри введені. І підраховує кількість цифр.

1. Завантажте систему візуального програмування Delphi.
2. Задайте заголовок, розмір та колір форми.
3. Вставити у форму компонент Edit. Витерти значення Edit1.
4. Вставте у форму текстові поля Label1-Label6 як показано на рис. 7.1 і надайте їм відповідні властивості відповідно до рис. 7.2.
5. Вставити у форму кнопку.

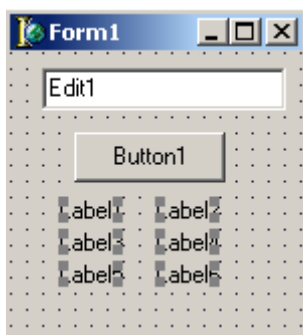


Рис. 7.1

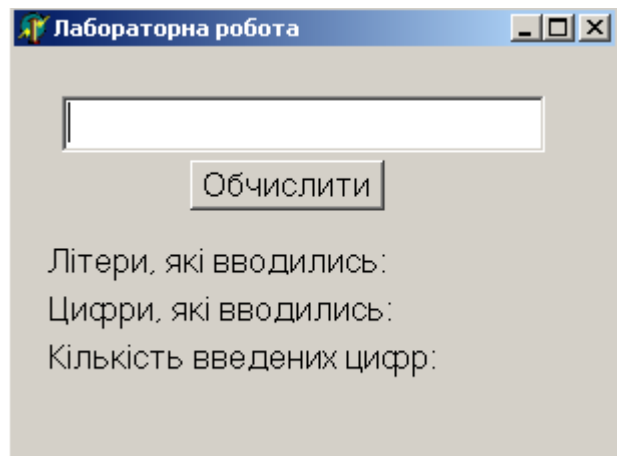


Рис. 7.2

Запрограмуймо кнопку «Обчислити»:

```
procedure TForm1.Button1Click(Sender: TObject);
var st:string;
    w,c,k,b:char;
    i,h:integer;
    lett:set of 'A'..'z'; //оголошується множина від А до z
    numb:set of '0'..'9'; //оголошується множина від 0 до 9

begin
Label2.Caption:=""; //очищується поле Label2
Label4.Caption:=""; //очищується поле Label4
lett:=[]; //онулюється множина lett
numb:=[]; //онулюється множина numb
st:=edit1.Text;
```

```

for i:=1 to Length(st) do //виконувати цикл до кінця стрічки
begin
if st[i] in ['0'..'9'] then //якщо символ стрічки належить до множини
begin
c:=st[i]; //символ записується у змінну c
numb:=numb+[c] //змна c додається до множини numb
end
else
w:=st[i]; //у іншому випадку всі символи записуються у змінну w
lett:=lett+[w];
end;

//вивести у Lable4 символи множини numb
for k:='0' to '9' do
if k in numb then
label4.Caption:=Label4.Caption+k;

//вивести у Lable6 символи множини lett
for b:='A' to 'z' do
if b in lett then
label2.Caption:=Label2.Caption+b;

//підрахунок кількості цифр у стрічці
h:=0;
for i:=1 to Length(st) do
if st[i] in ['0'..'9'] then h:=h+1;
label6.Caption:=FloatToStr(h);
end;

```

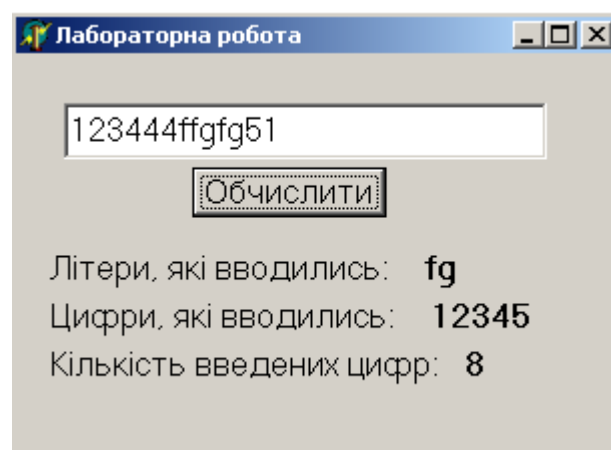


Рис. 7.3

7. Запустіть програму на виконання.
8. На рис.3 приведено виконання програми.
9. Створіть ехе-файл. Виконайте команду головного меню Project –

Build All (Сконструювати Все).

10. Продемонструйте створену форму викладачу.

Вимоги до оформлення звіту

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке множини.
2. Як анулюються множини.
3. Яким чином очищаються множини.
4. Як підрахувати кількість цифр у стрічці.
5. Як вивести символи множини lett?
6. Як вивести символи множини numb?
7. Як підрахувати кількість цифр у стрічці?

Практична робота № 8.

Тема: Програмування алгоритмів з використанням одновимірних масивів.


Мета: оволодіти практичними навиками роботи з масивами та особливостями організації циклічних процесів з заданим числом повторень.

Хід роботи.

1. Завантажте систему візуального програмування Delphi.
2. Задайте заголовок, розмір та колір форми. Для цього у вікні Object Inspector уведіть значення наступних властивостей: Caption (Заголовок форми) – «Практична робота», колір та розмір форми задайте на свій вибір.
3. Збережіть створену програму у власній папці. Для цього виберіть команду Save All (Зберегти все) головного меню File. Задайте імена програмного модуля та проекту.
4. Вставте у форму текстові поля Label1-Label2 як показано на рис.1 і надайте їм відповідні властивості. Для цього клікніть мишею на піктограмі Label на закладці Standard з палітри компонентів, а потім – у тому місці на формі, де має знаходитись текстове поле. Активізуйте це поле і у вікні Object Inspector задайте властивість Caption: для Label1 – “Сума:”, а для Label2 просто витерти і залишити порожнє (рис. 8.2).
5. Компонент StringGrid використовується для відображення інформації у вигляді таблиці. Таблиця складається з двох зон – фіксованої та робочої.

Фіксована зона призначена для виводу найменувань стрічок та стовпчиків робочої зони. Фіксована зона виділена іншим кольором і в неї заборонено вводити інформацію з клавіатури. Кількість стрічок та стовпчиків фіксованої зони встановлюється у властивостях FixedRows та FixedCols, відповідно.

Робоча зона складається з RowCount стрічок та ColCount стовпчиків інформації, яку можна міняти як програмно, так і за допомогою “миші” або клавіатури. Нумерація у таблиці починається з нуля.

Піктограма  компонента StringGrid знаходиться на сторінці Additional палітри компонентів. Розташуйте компонент відповідно рис. 8.1. Так як у нашому завданні для усіх компонентів StringGrid фіксована зона не використовується, то у інспекторі об'єктів значення властивостей FixedCols та FixedRows встановіть рівним 0. У завданні

використовується одномірний масив, тому встановіть значення кількості стрічок 1: RowCount=1, а стовпчиків замість n – відповідну кількість згідно свого варіанту: ColCount= n (наприклад, першого варіанту = 15)

По замовчуванні у компонент StringGrid заборонено ввід інформації з клавіатури, тому необхідно у інспекторі об'єктів двічі клікнути “мишею” на символі + біля властивості Options і у списку, що відкриється, встановити значення goEditing – True.

6. Вставте у форму кнопку, як показано на рис. 8.1, і надайте їй назву – «Обчислити».

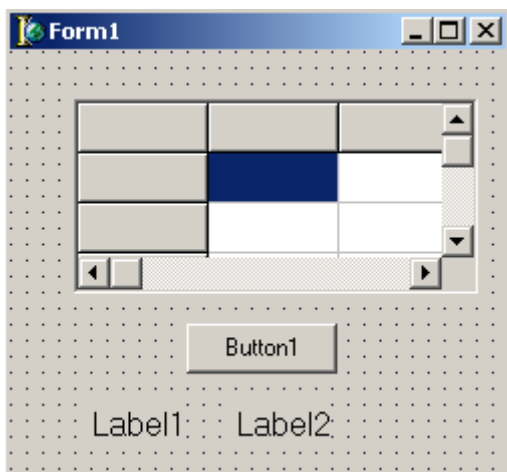


Рис. 8.1

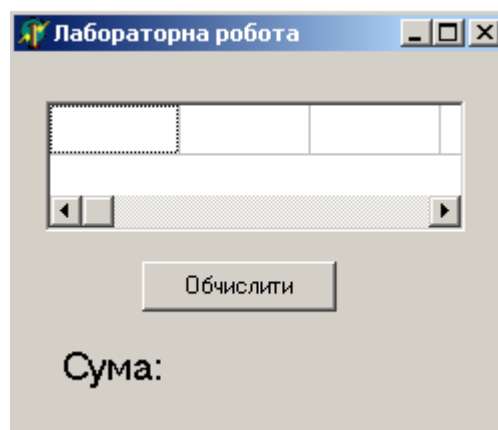


Рис. 8.2

7. Збережіть роботу на диску.

8. Далі потрібно розглянути приклад рішення завдання: визначити в одномірному масиві $x(3)$ суму парних елементів. Цей приклад НЕ ПЕРЕПИСУВАТИ, а лише РОЗГЛЯНУТИ!! І на його основі написати СВОЮ програму, згідно ВАШОГО варіанту (варіанти індивідуальних завдань приведені в кінці лабораторної роботи). У фігурних дужках {} записані коментарі:

Приклад:

Запрограмуймо кнопку «Обчислити»:

```
procedure TForm1.Button1Click(Sender: TObject);
const n=3;           {встановлюємо константу n}
var x:array[1..n] of real; {оголошуємо масив від 1 до n}
    s:real;
    i:integer;
begin
    s:=0;           {онулюємо змінну s}
    for i:=1 to n do
```

```

begin
    x[i]:=StrToFloat(StringGrid1.Cells[i-1,0]);      {присвоюємо зміній x
значення першого елемента масиву}
    if not odd(i) then s:=s+x[i];                  {якщо i не непарне, тоді
додати до s значення елемента x[i]}
end;
label2.Caption:=FloatToStr(s);                    {вивесли у Label2 значення
s, тобто результат}
end;

```

9. Після того, як ви складете програму ЗГІДНО ВАШОГО варіанту, запустіть програму на виконання. У компонент **StringGrid** введіть довільним чином елементи масиву. Після введення натисніть кнопку „Обчислити” та переконайтесь, що додаток функціонує правильно.

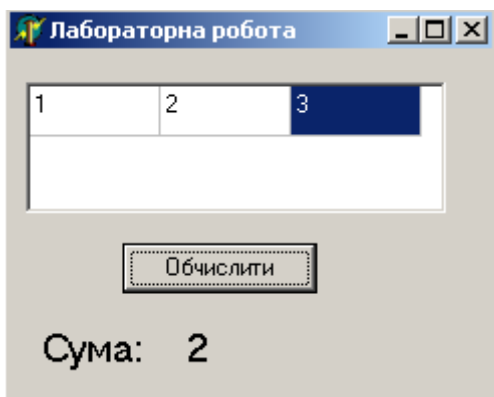


Рис. 8.3

На Рис. 8.3 приведено виконання програми для прикладу, що був розглянутий у попередньому пункті. Так як у цьому прикладі у масиві лише 3 елемента, то парних елементів залишається всього один. Тому і сума парних елементів дорівнює другому елементу масиву.

10. Збережіть програму.

11. Створіть exe-файл. Виконайте команду головного меню Project – Build All (Сконструювати Все).

12. Продемонструйте створену форму викладачу.

Індивідуальні завдання:

№	Масив	Завдання
1	x[15]	Знайти суму елементів масиву
2	y[10]	Знайти добуток елементів масиву
3	z[12]	Знайти значення найбільшого елемента масиву
4	x[11]	Знайти значення найменшого елемента масиву
5	z[12]	Знайти кількість додатних елементів масиву
6	a[11]	Знайти добуток відмінних від нуля елементів
7	b[14]	Знайти суму додатних елементів
8	c[16]	Знайти суму елементів масиву з непарними номерами

9	d[13]	Знайти кількість від'ємних елементів масиву
10	x[10]	Знайти середнє арифметичне елементів масиву
11	z[12]	Знайти добуток елементів з парними номерами
12	g[15]	Знайти кількість нульових елементів масиву
13	h[8]	Знайти суму від'ємних елементів масиву
14	w[11]	Знайти порядковий номер максимального елемента
15	x[14]	Знайти порядковий номер мінімального елемента

Вимоги до оформлення звіту:

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке масив?
2. Для чого використовується компонент StringGrid?


ПРАКТИЧНА РОБОТА №9.

Тема: Програмування алгоритмів з двовимірними масивами.

Мета: оволодіти практичними навиками роботи з двовимірними масивами та особливостями організації циклічних процесів з заданим числом повторень.

Хід роботи.

1. Завантажте систему візуального програмування Delphi.
2. Задайте заголовок, розмір та колір форми.
3. Вставте у форму текстові поля Label1-Label2 як показано на рис.1 і надайте їм відповідні властивості: для Label1 – “ Результат ”, а для Label2 просто витерти і залишити порожнє (рис. 2).
4. Розмістіть два компонента StringGrid відповідно до рис. 9.1.

Піктограма  компонента StringGrid знаходиться на сторінці Additional палітри компонентів. У інспекторі об'єктів значення властивостей FixedCols та FixedRows встановіть рівним 0 для StringGrid1 та StringGrid2.

У завданні використовується двовимірний масив, тому для StringGrid 1 встановіть значення кількості стрічок замість m – відповідну кількість згідно свого варіанту: RowCount= m , і стовпчиків замість n – відповідну кількість згідно свого варіанту: ColCount= n (наприклад, першого варіанту $m = 5, n = 5$).

StringGrid2 буде виводити результат у вигляді одномірному масиву. Тому встановіть значення кількості стрічок замість m – відповідну кількість згідно свого варіанту: RowCount= m , а стовпчиків: ColCount=1 (наприклад, першого варіанту $m = 5$).

По замовчуванню у компонент StringGrid заборонено ввід інформації з клавіатури, тому для StringGrid1 та StringGrid2 необхідно у інспекторі об'єктів двічі клікнути “мишею” на символі + біля властивості Options і у списку, що відкриється, встановити значення goEditing – True.

5. Вставте у форму кнопку, як показано на рис. 9.1, і надайте їй назву – «Обчислити».

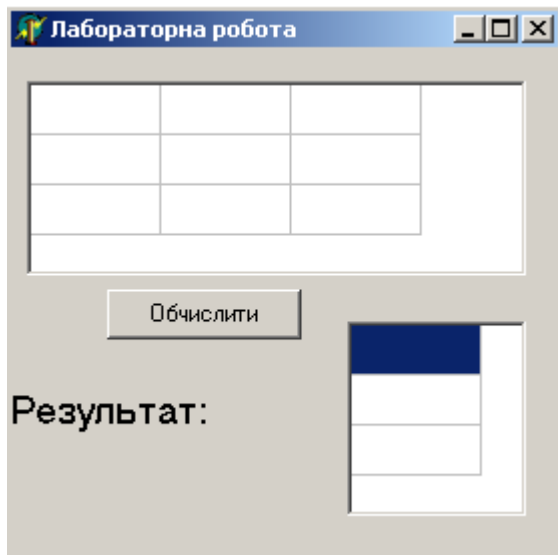


Рис. 9.1

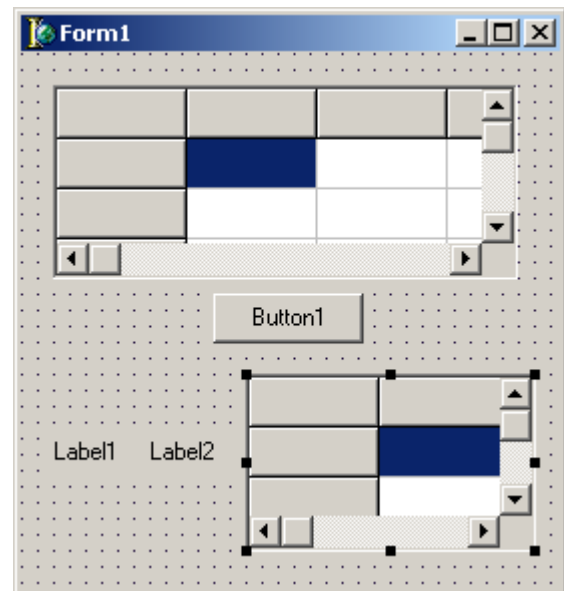


Рис. 9.2

6. Далі потрібно розглянути приклад рішення завдання: знайти суму елементів масиву $x(3,3)$; знайти суму елементів кожної стрічки матриці. Цей приклад НЕ ПЕРЕПИСУВАТИ, а лише РОЗГЛЯНУТИ!! І на його основі написати СВОЮ програму, згідно ВАШОГО варіанту (варіанти індивідуальних завдань приведені в кінці лабораторної роботи).

ПРИКЛАД:

Запрограмуймо кнопку

«Обчислити»:

```
procedure
```

```
TForm1.Button1Click(Sender:
```

```
TObject);
```

```
const n=3;
```

```
      m=3;
```

```
var x:array[1..n,1..m] of real;
```

```
    s:array[1..n] of real;
```

```
    y:array[1..n] of real;
```

```
    i,j:integer;
```

```
    b:real;
```

```
begin
```

```
  begin
```

```
    b:=0;
```

```
    for i:=1 to n do
```

```
      for j:=1 to m do
```

```
        begin
```

```

x[i,j]:=StrToFloat(StringGrid1.Cells[i-
1,j-1]);
    b:=b+x[i,j];
    end;
    label2.Caption:=FloatToStr(b);
end;
begin
for i:=1 to n do
begin
s[i]:=0.0;
for j:=1 to m do
begin

```

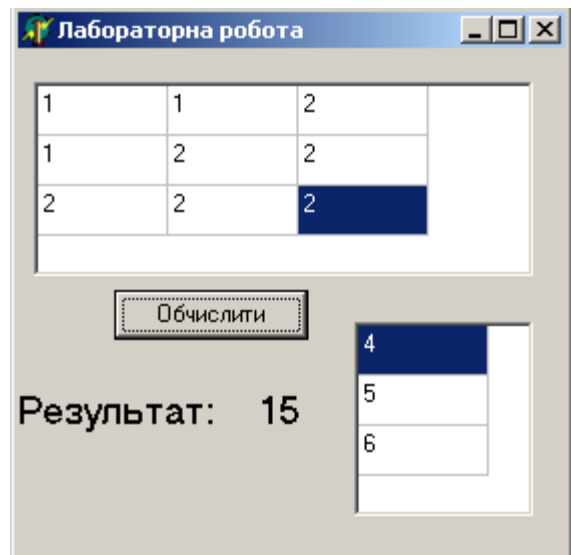


Рис. 9.3

```

x[i,j]:=StrToFloat(StringGrid1.Cells[i-
1,j-1]);
    s[i]:=s[i]+x[i,j];
    StringGrid2.Cells[0,i-
1]:=FloatToStrF(s[i],ffFixed,4,0);
    end;
end;
end;
end;

```

7. Після того, як ви складете програму ЗГІДНО ВАШОГО варіанту, запустіть програму на виконання. У компонент StringGrid введіть довільним чином елементи масиву. Після введення натисніть кнопку „Обчислити” та переконайтесь, що додаток функціонує правильно.

8. На рис. 9.3 приведено виконання програми для прикладу, що був розглянутий у попередньому пункті.

9. Збережіть програму.

10. Створіть ехе-файл. Виконайте команду головного меню Project – Build All (Сконструювати Все).

11. Продемонструйте створену форму викладачу.

Індивідуальні завдання

№	Масив	Завдання №1	Завдання №2
1	x[5,5]	Знайти суму елементів масиву	Знайти суму елементів кожної стрічки матриці
2	y[4,4]	Знайти добуток елементів масиву	Знайти суму додатнієї елементів кожної стрічки
3	z[5,4]	Знайти значення найбільшого елементу масиву	Знайти добуток елементів кожної стрічки

4	x[4,5]	Знайти значення найменшого елемента масиву	Знайти добуток елементів кожного стовбця
5	z[6,3]	Знайти кількість додатніх елементів масиву	Знайти для кожного рядка суму елементів непарних стовпців
6	a[3,5]	Знайти добуток відмінних від нуля елементів	Знайти середнє арифметичне кожного рядка
7	b[4,6]	Знайти суму додатніх елементів	Знайти суму елементів кожної стрічки
8	c[3,4]	Знайти суму елементів масиву, більших 10	Знайти максимальний елемент кожної стрічки
9	d[4,3]	Знайти кількість від'ємних елементів масиву	Знайти мінімальний елемент кожної стрічки
10	x[4,4]	Знайти середнє арифметичне елементів масиву	Знайти суму елементів більших 10 кожної стрічки
11	z[6,6]	Знайти добуток від'ємних елементів	Знайти добуток елементів непарних стовбців
12	g[6,4]	Знайти кількість нульових елементів масиву	Знайти суму максимальних елементів кожної стрічки
13	h[3,5]	Знайти суму від'ємних елементів масиву	Знайти кількість від'ємних елементів кожного стовбця
14	w[4,5]	Знайти номер стовбця та рядка в якому знаходиться максимальний елемент	Знайти кількість додатніх елементів кожного стовбця.
15	x[5,5]	Знайти номер стовбця та рядка в якому знаходиться мінімальний елемент	Знайти кількість елементів більших по модулю 3.14 для кожного стовбця.

Вимоги до оформлення звіту:

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що являє собою значення RowCount?
2. Що являє собою значення ColCount?
3. Що являє собою значення FixedCols?
4. Що являє собою значення FixedRows?
5. Як створити exe-програму?
6. Як зберегти програму ?
7. Чому значення властивостей FixedCols та FixedRows встановлюють рівним 0 для StringGrid1 та StringGrid2?

Практична робота №10.

Тема: Програмування алгоритмів з використанням символічних рядків.

Мета: освоїти вживання компонентів ListBox і ComboBox і створити програму-додаток, в якій використовуються рядки.

Теоретичні відомості.

1. Приклад створення рядка.

Завдання:

Створити Windows-додаток для підрахунку кількості слів в довільному рядку. Слова в рядку розділяються будь-якою кількістю пропусків. Введення рядка закінчувати натисненням клавіші Enter. Робота додатку повинна завершуватися натисненням кнопки Close.

Один з можливих варіантів панелі інтерфейсу створюваного програмного-додатку показаний на рис 10.1.

2. Розміщення компонентів на Формі.

При роботі з рядками введення і виведення інформації на екран зручно організовувати за допомогою компонентів ListBox і ComboBox.

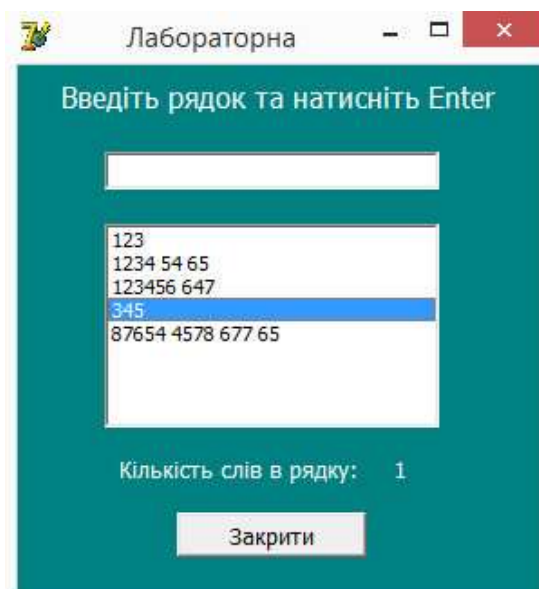


Рис. 10.1

Компонент **ListBox** є списком, елементи якого вибираються за допомогою клавіатури або “миші”. Список елементів задається властивістю Items, методи Add, Delete і Insert якої використовуються для додавання, видалення і вставки рядків, відповідно. Для визначення номера виділеного елементу використовується властивість ItemIndex.

Компонент **ComboBox** є комбінацією списку ListBox і редактора Edit, тому практично всі властивості запозичають в цих компонентів.

Для роботи з вікном редагування використовується властивість Text як в Edit, а для роботи із списком вибору використовується властивість Items як в ListBox. Існує 5 модифікацій компонента, визначених його властивістю Style. В модифікації csSimple список завжди розкритий, в інших він розкривається після натиснення кнопки праворуч від редактора.

Компоненти ListBox і ComboBox знаходяться на сторінці Standard Палітри Компонентів.

Компонент **BitBtn** розташований на сторінці Additional Палітри Компонентів і є різновидом стандартної кнопки Button. Його відмітна особливість – наявність зображення на поверхні кнопки, яка визначається властивістю Glyph. Крім того, є властивість Kind, яка задає одну з 11 стандартних різновидів кнопок. Натиснення будь-якої з них, окрім bkCustom і bkHelp закриває модальне вікно. Кнопка bkClose закриває головне вікно і завершує роботу програми.

3. Створення процедур обробки подій

У момент запуску програми- додатку, коли панель інтерфейсу з'являється на екрані, для користувача зручно, щоб курсор вже знаходився в полі редактора компонента ComboBox. При активізації її форми виникає подія OnActivate, яку можна використовувати для передачі фокусу введення компоненту ComboBox. Для створення процедури-обробника цієї події необхідно в інспекторі об'єктів вибрати компонент Form1, на сторінці Events знайти подію OnActivate і двічі клацнути “мишею” по його правій (білої) частині. Курсор встановиться в тексті процедури-обробника події активізації Форми:

```
procedure TForm1.FormActivate(Sender: TObject).
```

В цьому місці процедури наберіть оператор передачі фокусу введення компоненту ComboBox1 (див. текст модуля UnStr, який приведений в п. 2).

Відповідно до завдання необхідно, щоб при натисненні клавіші Enter рядок символів, який користувач набрав в полі редагування, переносився в список вибору компонента **ComboBox**. Для створення процедури-обробника цієї події необхідно в інспекторі об'єктів вибрати компонент ComboBox1, на сторінці Events знайти подію OnKeyPress і двічі клацнути “мишею” по його правій частині. Курсор встановиться в тексті процедури-обробника події натиснення клавіші на клавіатурі:

```
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char).
```

В цьому місці процедури, користуючись текстом модуля UnStr, наберіть оператори, які при натисненні клавіші Enter переносять рядок з поля редагування в список вибору і очищують поле редагування.

Процес створення процедури-обробника події натиснення клавіші “миші” в списку вибору

```
procedure TForm1.ComboBox1Click(Sender: TObject)
```

виконується аналогічно для події OnClick компоненту ComboBox1. Користуючись текстом модуля UnStr, наберіть оператори, які здійснюють основний алгоритм обробки символів вибраного рядка.

4. Текст модуля UnStr:

```
Unit UnStr;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs
  StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    Label2: TLabel;
    Label3: TLabel;
    BitBtn1: TBitBtn;
    ComboBox1: TComboBox;
    Label1: TLabel;
  procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
  procedure ComboBox1Click(Sender: TObject);
  procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;

Implementation
{$R *.DFM}

// Обробка події активізації Форми
procedure TForm1.FormActivate(Sender: TObject);
begin
  ComboBox1.SetFocus;           // передача фокусу введення ComboBox1
end;
```

```

// Обробка події введення символу і натиснення клавіші Enter
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
  if key=#13 then // якщо натискання клавіші Enter то
    begin // рядок з поля редагування заноситься
      ComboBox1.Items.Add(ComboBox1.Text); // в список вибору
      ComboBox1.Text:=""; // очищення вікна редагування
    end;
end;

```

```

// Обробка події натиснення клавіші "миші" в списку вибору
procedure TForm1.ComboBox1Click(Sender: TObject);
var
  -й : string;
  n,i,nst,ind: integer;
begin
  n:=0; // n містить кількість слів
  ind:=0;
  nst:=ComboBox1.ItemIndex; // визначення номера вибраного рядка
  -й:=ComboBox1.Items[nst]; // привласнюється вибраний рядок
  for i:=1 to Length(-й) do // перегляд всіх символів рядка
    case ind
      0 : if -й[i]<>' ' then // якщо зустрівся символ
        begin
          ind:=1;
          n:=n+1; // кількість слів збільшується на одиницю
        end;
      1 : if -й[i]=' ' then // якщо зустрівся пропуск
        ind:=0;
    end;
end;
Label3.Caption:=IntToStr(n); // виведення кількості слів в Label3
end;
end.

```

5. Виконання індивідуального завдання:

У всіх завданнях початкові дані вводити за допомогою компоненту Edit в компонент ListBox, або за допомогою властивості Text у властивість Items компоненту ComboBox. Результат виводити за допомогою компоненту Label. Введення рядка закінчувати натисненням клавіші Enter. Робота програми-додатку повинна завершуватися натисненням кнопки Close.

Для перевірки функціонування програми-додатку підготувати декілька тестів.

Індивідуальні завдання:

1. Даний рядок, що складається з груп нулів і одиниць. Кожна група відділяється один від одного одним або декількома пропусками. Знайти кількість груп з п'ятьма символами.

2. Даний рядок, що складається з груп нулів і одиниць. Кожна група відділяється один від одного одним або декількома пропусками. Знайти і вивести на екран найкоротшу групу.

3. Даний рядок, що складається з груп нулів і одиниць. Кожна група відділяється один від одного одним або декількома пропусками. Підрахувати кількість символів в щонайдовшій групі.

4. Даний рядок, що складається з груп нулів і одиниць. Кожна група відділяється один від одного одним або декількома пропусками. Знайти і вивести на екран групи з парною кількістю символів.

Вимоги до оформлення звіту :

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке компонент ListBox?
2. Що таке компонент ComboBox?
3. Властивості компонента ListBox?
4. Властивості компонента ComboBox?
5. Як створюється процедура обробки подій?
6. Що таке компонент BitBtn?
7. Властивості компонента BitBtn?
8. Як збільшити кількість слів на одиницю?
9. Як визначити номери вибраного рядка?

Практична робота №11.

Тема: Програмування алгоритмів використанням записів.

Мета: створити програму для використання даних для запису.

Теоретичні відомості.

Запис – це структура даних, що складається з фіксованої кількості компонентів, званих полями запису. На відміну від масиву компоненти (поля) запису можуть бути різного типу. Щоб можна було посилатися на той або інший компонент запису, поля іменуються.

Структура оголошення типу запису така:

`<ім'я типу> = record <список полів> end;`

Тут `<ім'я типу>` – правильний ідентифікатор; `record / end` – зарезервовані слова *{запис, кінець}*; `< список полів >` – список полів; є послідовністю *розділів запису*, між якими ставиться крапка з комою.

Кожний розділ запису складається з одного або декількох ідентифікаторів полів, відокремлюваних один від одного комами. За ідентифікатором (ідентифікаторами) ставиться двокрапка і опис типу поля (полів), наприклад:

```
type
  Birthday = record Day, Month: Byte;
  Year : Word end;
var
  a,b : Birthday;
```

У даному прикладі тип `Birthday` (*день народження*) є запис з полями `Day`, `Month` і `Year` (*день, місяць і рік*); змінні `a` і `b` містять записи типу `Birthday`.

Як і в масиві, значення змінних типу запису можна привласнювати іншим змінним того ж типу, наприклад:

`a := b;`

До кожного з компонентів запису можна дістати доступ, якщо використати складове ім'я, тобто вказати ім'я змінної, потім крапку і ім'я поля:

`a.day := 27;`
`b.year := 1939;`

Для вкладених полів доводиться продовжувати уточнення:

```
type
  Birthday = record Day,Month: Byte;
  Year : Word end;
var
  z : record
```

```
Name : String;  
Bd : Birthday end;  
begin  
if c.Bd.Year = 1989 then ... end.
```

Щоб спростити доступ до полів запису, використовується оператор приєднання with:

```
with <переменная> do <оператор>;
```

Тут with, do - зарезервовані слова (з, робити);

<переменная> - ім'я змінної типу запис, за яким, можливо

слідє список вкладених полів; <оператор> – будь-який оператор

Object Pascal.

Наприклад:

```
c.Bd.Month := 9;
```

Це еквівалентно:

```
with z.Bd do Month := 9;
```

```
або with z do with Bd do Month := 9;
```

```
або with z, Bd do Month := 9;
```

Приклад створення програми.

Завдання:

Створити Windows-додаток для обробки відомості про успішність здобувачів в кількості 9 чоловік. Кожний запис повинен містити прізвище, ініціали, а також оцінки по фізиці, математиці і твору. Вивести список здобувачів, відсортований в порядку зменшення їх середнього балу.

Один з можливих варіантів панелі інтерфейсу створюваного програмного додатку показаний на рис.1.

1. Розміщення компонентів на Формі.

StringGrid

The screenshot shows a Windows form titled 'Form1' with a blue border. Inside the form is a table with 6 columns: '№пп', 'Прізвище, ініціали', 'Математика', 'Фізика', 'Твір', and 'Ср.бал'. The table contains 9 rows of data. A blue highlight is visible on the 'Твір' column of the first row. Below the table is a button labeled 'Сортувати'. A line points from the text 'StringGrid' to the table.

№пп	Прізвище, ініціали	Математика	Фізика	Твір	Ср.бал
1	Восьмий В.В.	5	5	5	5,00
2	Сьомий С.С.	5	5	4	4,67
3	Шостий Ш.Ш.	5	4	4	4,33
4	Дев'ятий Д.Д.	3	5	5	4,33
5	Четвертий Ч.Ч.	4	4	4	4,00
6	П'ятий П.П.	3	4	5	4,00
7	Третій Т.Т.	3	4	4	3,67
8	Другий В.В.	3	3	4	3,33
9	Перший П.П.	3	3	3	3,00

При роботі з записами введення і виведення інформації на екран зручно організовувати за допомогою компоненту StringGrid.

У цьому завданні для нанесення відповідних написів в колонках і рядках використовується фіксована зона компоненту StringGrid, тому в інспекторі об'єктів значення властивостей FixedCols і FixedRows встановіть рівними 1. Відповідно до завдання встановіть значення властивості ColCount=6, а значення властивості RowCount=10. Для можливості перегляду всього списку здобувачів в компоненті StringGrid зручно використовувати вертикальну лінійку прокрутки, тому встановіть властивість ScrollBars в ssVertical. Відкрийте список опцій властивості Options і встановіть значення goEditing в True – це дасть можливість редагувати інформацію в компоненті StringGrid за допомогою клавіатури і “миші”.

2. Створення процедур обробки подій FormCreate і Button1Click.

Подвійним натисненням клавіші “миші” на формі і кнопці Button1 створіть відповідні процедури обробки подій. Використовуючи текст модуля UnZap, уважно наберіть операції Рис. 1 процедур.

3. Текст модуля UnZap:

```
Unit UnZap;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs
  StdCtrls, Buttons, Grids;
type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    Button1: TButton;
  procedure FormCreate(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
  {$R *.DFM}
type
  zap=record           // оголошення запису
```

```

    fio      :string[20];
    mat,fiz,soch:integer;
    srbal    :extended
end;
var
MZap:array[1..9] zap; // оголошення масиву записів
procedure TForm1.FormCreate(Sender: TObject);
var
i:integer;
begin
    with StringGrid1 do
    begin // занесення інформації в середину StringGrid1
        Cells[0,0]:='№пп';
        Cells[1,0]:='Фамилия,инициалы';
        Cells[2,0]:='Математика';
        Cells[3,0]:='Физика';
        Cells[4,0]:='Сочинение';
        Cells[5,0]:='Ср.балл';
        for i:=1 to 9 do
            Cells[0,i]:=IntToStr(i);
        Cells[1,1]:='Первый П.П.'; Cells[2,1]:='3'; Cells[3,1]:='3'; Cells[4,1]:='3';
        Cells[1,2]:='Второй В.В.'; Cells[2,2]:='3'; Cells[3,2]:='3'; Cells[4,2]:='4';
        Cells[1,3]:='Третий Т.Т.'; Cells[2,3]:='3'; Cells[3,3]:='4'; Cells[4,3]:='4';
        Cells[1,4]:='Четвертый Ч.Ч.'; Cells[2,4]:='4'; Cells[3,4]:='4'; Cells[4,4]:='4';
        Cells[1,5]:='Пятый П.П.'; Cells[2,5]:='3'; Cells[3,5]:='4'; Cells[4,5]:='5';
        Cells[1,6]:='Шестой Ш.Ш.'; Cells[2,6]:='5'; Cells[3,6]:='4'; Cells[4,6]:='4';
        Cells[1,7]:='Седьмой С.С.'; Cells[2,7]:='5'; Cells[3,7]:='5'; Cells[4,7]:='4';
        Cells[1,8]:='Восьмой В.В.'; Cells[2,8]:='5'; Cells[3,8]:='5'; Cells[4,8]:='5';
        Cells[1,9]:='Девятый Д.Д.'; Cells[2,9]:='3'; Cells[3,9]:='5'; Cells[4,9]:='5';
        for i:=1 to 9 do
            with MZap[i] do
            begin // формування полів масиву записів
                fio:=Cells[1,i];
                mat:=StrToInt(Cells[2,i]);
                fiz:=StrToInt(Cells[3,i]);
                soch:=StrToInt(Cells[4,i]);
                srbal:=(mat+fiz+soch)/3; // обчислення значення
                середнього бала
                Cells[5,i]:=FloatToStrF(srbal,ffFixed,5,2); // виведення значення
                середнього балла
            end; // в останню колонку StringGrid1
        end;
    end;
end;
procedure TForm1.Button1Click(Sender: TObject);
var

```

```

i,j :integer;
vper:zap;
begin
for i:=1 to 9 do
with StringGrid1,MZap[i] do
begin
fio:=Cells[1,i];
mat:=StrToInt(Cells[2,i]);
fiz:=StrToInt(Cells[3,i]);
soch:=StrToInt(Cells[4,i]);
srbal:=(mat+fiz+soch)/3;
Cells[5,i]:=FloatToStrF(srbal,ffFixed,5,2);
end;
for i:=2 to 9 do // сортування методом "пухирця"
for j:=9 downto i do
if MZap[j-1].srbal<MZap[j].srbal then
begin
vper:=MZap[j-1];
MZap[j-1]:=MZap[j];
MZap[j]:=vper;
end;
for i:=1 to 9 do // заповнення осередків
StringGrid1 полями масиву записів
with StringGrid1,MZap[i] do
begin
Cells[1,i]:=fio;
Cells[2,i]:=IntToStr(mat);
Cells[3,i]:=IntToStr(fiz);
Cells[4,i]:=IntToStr(soch);
Cells[5,i]:=FloatToStrF(srbal,ffFixed,5,2);
end;
end;
end.

```

4. Виконання індивідуального завдання.

По вказівці викладача виберіть своє індивідуальне завдання. Створіть програму з записом-додатку і протестуйте його роботу.

Індивідуальні завдання:

1. Поля шахівниці характеризуються записом:

```
Type  
Pole=record  
    Ver: 1..8;           {вертикальні координати}  
    Hor: (a,b,c,d,e,f,g,h); {горизонтальні координати}  
end;
```

Вивести шахівницю, помітивши хрестиками всі поля, які «б'є» ферзь, що стоїть на полі з координатами *Veri* і *Hori*, і нулями вся решта полів.

2. Поля шахівниці характеризуються записом (див. завдання 1)

```
Var Figura:Pole;
```

Вивести повідомлення чи може кінь за один хід перейти з поля *Figurai* на поле *Figuraj*.

3. Type Karta=record

```
    m: (piki,trefi,bubni,chervi);           {масть}  
    d:(shest,sem,vosem,devjat,desjat,valet,dama,korol,tuz); {гідність}  
end;  
Var k1,k2:Karta;
```

Вивести повідомлення чи «б'є» карта *k1*, карту *k2*, з урахуванням того, що масть є козирем.

4. У магазині формується список осіб, що записалися на покупку товару підвищеного попиту. Кожний запис цього списку містить: порядковий номер, Ф.І.О., домашню адресу покупця і дату постановки на облік. Видалити з списку всі повторні записи, перевіряючи Ф.І.О. і домашню адресу.

Вимоги до оформлення звіту :

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке запис?
2. З чого складається запис?
3. Яка структура оголошення типу запису?
4. Як дістати доступ до компонентів запису?
5. Як відбувається сортування методом пухирця?
6. Як відбувається формування полів масиву записів?
7. Як заповнюються осередки `StringGrid1` полями масиву записів?

Практична робота № 12.

Тема: Використання компонентів типу TLabel, TEdit, TMemo для вводу/виводу даних.

Мета: Застосовуючи компоненти TLabel, TEdit, TMemo виконати завдання на лабораторну роботу та оформити звіт.

Теоретичні відомості.

1. Компонент типу TLabel – це мітка, тобто елемент, що містить статичний текст, який користувач програми змінити не може. Такі компоненти служать для виводу напису у потрібному місці вікна. Тобто мітку можна використати, як засіб для виводу результату роботи певної підпрограми.

Компонент мітка знаходиться на закладці **Standard** палітри компонентів. Для виводу напису служить властивість **Caption**, яка доступна через інспектор об'єктів для задання напису на етапі створення інтерфейсу програми.

Властивість Caption – відображає дані стрічкового типу. Зовнішній вигляд напису можна задати через властивість – *Font* (шрифт), яка містить ряд підвластивостей: *Size* (розмір шрифту) натуральне число *Color* (колір шрифту) одна з системних констант Windows, що задає колір (можна використати числовий еквівалент константи). *Name* (назва шрифту) та ін.

2. Компонент типу TEdit – поле однострічкового редактора, використовується як для вводу, так і для виводу текстових даних у вигляді однієї стрічки. Стрічку містить властивість **Text**. Компонент однострічкового редактора знаходиться на закладці **Standard** палітри компонентів. Можливості форматування у цьому компоненті такі ж, як і у компонента **типу TLabel**. Максимальну довжину стрічки задає властивість **MaxLength**. Якщо ця властивість рівна 0, то довжина стрічки не обмежується. Можливістю зміни (редагування) тексту у полі однострічкового редактора керує властивість **Enabled** логічного типу.

3. Компонент типу TMemo служить для вводу/виводу тексту в полі багатострічкового редактора на подоби до блокноту. Цей компонент може містити текст розміром до 64К.

Компонент багатострічкового редактора знаходиться на закладці **Standard** палітри компонентів.

Для зміни тексту через інспектор об'єктів служить **властивість Strings** – подвійне клацання мишею, яка відкриває вікно редагування вмістимого компоненту типу **TMemo**.

Для програмної зміни вмістимого цього компоненту служить властивість `Lines`, яка містить ряд методів, зокрема:

- **Clear** – (очистка) очистка вмістимого поля `TMemo`.
- **Add** – (стрічка) додання стрічки до існуючих.
- **LoadFromFile** – (ім'я файлу) заповнити вмістиме компоненту текстом з файлу, ім'я якого вказане як аргумент модуля.
- **SaveToFile** – (ім'я файлу) записати вмістиме компоненту у файл, ім'я якого вказане як аргумент методу.

Оскільки у програмі може виникнути потреба обробляти числові дані, а згадані компоненти працюють тільки зі стрічками, то є потреба проводити взаємні перетворення між числами та стрічками.

Перетворення числа у стрічку:

цілого: функція `IntToStr` (ціле число) – повертає стрічку, яка є результатом перетворення цілого числа:

дробового: функція `FloatToStr` (дробове число) – повертає стрічку, яка є результатом перетворення дробового числа:

будь-якого: процедура `Str(N,S)` – перетворення числа `N` у стрічку `S`.

Перетворення стрічки у число:

ціле: функція `StrToInt` (стрічка) – повертає ціле число, яке є результатом перетворення стрічки:

дробове: функція `StrToFloat` (стрічка) – повертає дробове число, яке є результатом перетворення стрічки:

будь-яке: процедура `Val(S.V.C)` – перетворення стрічки `S` у число `V`. `C` – код помилки (номер символу, який неможливо перетворити). Якщо перетворення здійснено успішно, то `C=0`.

Хід роботи.

1. Запустити Delphi та розмістити на формі:

- зверху розмістити компонент мітка для виводу напису про призначення програми:
- зліва розмістити один під одним три компоненти поля однострічкового редактора для вводу даних:
- зліва біля кожного поля однострічкового редактора розмістити мітку для напису, яку змінну потрібно вводити – `X` чи `Y` та результат обчислення:

- зовнішній вигляд шрифту кожного з цих компонентів встановити через інспектор об'єктів на власний розсуд, використовуючи властивість `Font`:
- поле редактора, в якому виводиться результат, зробити недоступним для редагування. Для цього використати подію `OnCreate` форми – клацнути мишею по вільному місцю форми два рази, ввести у заголовку процедури, що появиться після цього, програмний код, який встановлює властивість `Enabled` відповідного компоненту `Edit` у властивість `false`.

Наприклад: цей рядок може виглядати так: `Edit3.Enabled:=false;`

- вирівняти компоненти так, щоб вони мали однакові розміри та мали однакові відступи один від одного:
- під компонентами типу `TEdit` розмістити дві кнопки. Змінити написи на них “Обчислити” та “Очистити”. Для цього по чергово для кожної кнопки виконати такі дії: виділити кнопку мишею, в інспекторі об'єктів змінити властивість `Caption`. Натискання першої кнопки повинно викликати початок процесу обчислення з даними, введеними у два перші поля `Edit` та вивід результату у третє однострічкове поле. Натискання другої кнопки повинно викликати очистку вмісту всіх полів з даними:
- справа на формі розмістити компонент `Мемо` і використовуючи інспектор об'єктів, очистити його вміст від будь-яких написів:
- під компонентом `Мемо` розмістити дві кнопки з написами “Добавити” та “Очистка” по аналогії до того, як це було зроблено з кнопками під полями для вводу даних. Натискання першої кнопки повинно викликати добавлення до поля `Мемо` стрічки з поля `Edit`, в якому виведено результат. Натискання другої кнопки має викликати витирання всього вмісту поля `Мемо`:
- внизу форми розмістити кнопку з написом “ Вихід”, натискання якої повинно викликати закриття вікна програми, та вихід з неї:
- коли на формі не поміщаються всі компоненти, розміри її змінити стандартним прийомом зміни розмірів вікна в ОС `Windows`.

2. Написати процедуру обчислення значення виразу згідно завдання , як обробку події натискання відповідної кнопки. Для цього виділити потрібну кнопку або два рази клацнути по ній, або в інспекторі об'єктів перейти на закладку *Events* і два рази клацнути мишею біля події *OnClick*. В будь-якому випадку появиться заготовка процедури. Ваше завдання використовуючи наведені теоретичні

відомості, організувати зчитування значень, введених у перші два поля однострічкових редакторів, у відповідні описані вами змінні, виконати потрібні операції згідно завдання та вивести результат у третє поле однострічкового редактора. При необхідності виконувати перетворення між числовими та стрічковими даними.

3. Написати процедуру очистки всіх однострічкових полів, як обробку полів натискання відповідної кнопки по аналогії до попереднього пункту. Тіло цієї процедури повинно складатися з трьох операторів присвоювання властивості *Text* кожного з полів Edit порожньої стрічки. Наприклад: `Edit1.Text=""`;

4. Написати процедуру додавання нової стрічки до поля Memo, як обробку події натискання відповідної кнопки. Тіло цієї процедури повинно складатися з одного оператора, який використовує метод додавання стрічки у поле Memo- до існуючих. В якості аргументу методу має використовуватися стрічка, що міститься у однострічковому полі призначеному для виводу результату.

5. Написати процедуру очистки поля *Memo*, як обробку події натискання відповідної кнопки. Ця процедура повинна містити оператор, що викликає метод витирання всіх стрічок.

6. Написати процедуру, яка здійснює вихід з програми, як обробку події натискання відповідної кнопки. Тіло цієї процедури має складатися з єдиного оператора *Close*; .

7. Запустити програму і в разі потреби справити всі помилки. Пред'явити робочу програму викладачу.

8. Оформити віт по виконанню лабораторної роботи.

Завдання:

Вести два числа та обчислити значення виразу згідно варіанту. Вивести результат:

Варіант	X	Y	Вираз для обчислення
1	25	65	$\frac{1}{X} + \sqrt{XY}$
2	2.30	0.25	$2X - \ln(Y)$
3	4	9	$e^X XY$
4	0	56	$\sin(X - Y) + \cos(3Y)$
5	9.71	1.25	$\sqrt{\sin(X - Y) + \cos(3Y)}$
6	345	654	$\sqrt{e^X XY \cdot \ln(X - Y)}$
7	102.30	30.20	$\frac{1}{2X} + \frac{X}{3Y^2}$

8	184	204	$\cos^2(XY) + \sin\left(\frac{Y}{X}\right)$
9	854	236	$e^{\cos^2(XY) + \sin\left(\frac{Y}{X}\right)}$
10	965.65	142.45	$\ln\left(\frac{1}{2X} + \frac{X}{3Y^2}\right)$

Вимоги до оформлення звіту.

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст процедур обробки подій натискання всіх кнопок.
4. Відповідь на контрольні запитання.

Контрольні питання.

1. Призначення компоненту типу TLabel та TEdit. Що в них спільного і що відмінного?
2. Як відформувати вигляд напису у компонентах типу TLabel та TEdit програмно та через інспектор об'єктів?
3. Як можна керувати вмістом поля типу TMemo на етапі конструювання програми та при її виконанні?
4. За допомогою яких методів можна прочитати у поле TMemo текст з файлу та записати вміст поля у файл?
5. Чому потрібно здійснювати перетворення між стрічковими та числовими даними при роботі з компонентами TLabel та TEdit та TMemo? Які засоби для цього існують?
6. Як вирівняти компоненти на формі таким чином, щоб вони були однакової висоти і відстані між ними по висоті теж були однакові?
7. За що відповідає властивість Caption компонента типу TLabel? Яка різниця між цією властивістю та властивістю Name?
8. За що відповідають властивості Top, Height, Width?

Практична робота №13

Тема: Програмування алгоритмів з використанням компонентів **StringGrid** при написанні програми.

Мета: освоїти вживання компоненту **StringGrid** і створити додаток, в якому використовуються масиви.

Теоретичні відомості.

1. Приклад створення додатку.

Завдання:

Створити Windows-додаток для обчислення вектору $x=\{x_1, x_2, \dots, x_m\}$, рівного p -й рядку матриці $A=\{a_{ij}\}$ ($x_j=a_{pj}$, $j=1, 2, \dots, m$) і вектора $y=\{y_1, y_2, \dots, y_n\}$, рівного q -му стовпцю матриці $A=\{a_{ij}\}$ ($y_i=a_{iq}$, $i=1, 2, \dots, n$) ($n \leq 6, m \leq 8$). У панелі інтерфейсу передбачити можливість управління розмірністю масивів.

Один з можливих варіантів панелі інтерфейсу створюваного додатку показаний на рис. 1.

2. Розміщення компонентів на Формі

При роботі з масивами введення і виведення інформації на екран зручно організовувати за допомогою компоненту **StringGrid**.

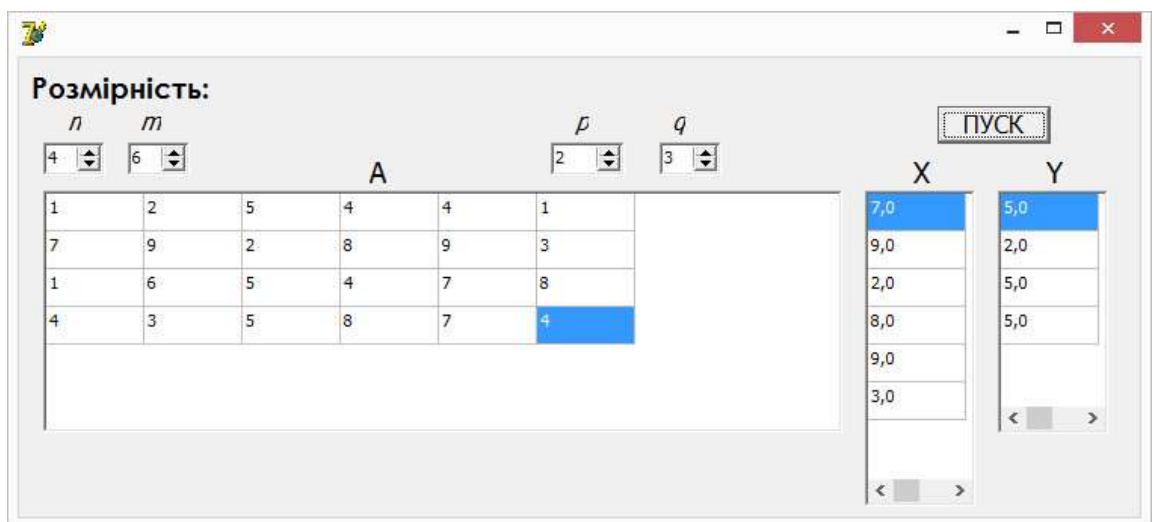



Рис. 13.1

Компонент **StringGrid** використовується для відображення інформації у вигляді таблиці. Таблиця містить дві зони – фіксовану і робочу. *Фіксована зона* служить для виводу найменувань рядків і стовпців робочої зони і управління їх розмірами за допомогою “миші”. Фіксована зона виділена іншим кольором і в неї заборонено введення

інформації з клавіатури. Кількість рядків і стовпців фіксованої зони встановлюється у властивостях `FixedRows` і `FixedCols`, відповідно.

Робоча зона містить `RowCount` рядків і `ColCount` стовпців інформації, яку можна змінювати як програмно, так і за допомогою “миші” або клавіатури.

Доступ до інформації в програмі здійснюється за допомогою властивості `Cells[ACol, ARow: integer]: string`, де `ACol` – номер стовпця, а `ARow` – номер рядка таблиці, причому нумерація починається з нуля.

Піктограма  компоненту `StringGrid` знаходиться на сторінці `Additional` Палітри Компонентів. Оскільки в нашому завданні для всіх компонентів `StringGrid` фіксована зона не використовується, в Інспекторі Об'єктів значення властивостей `FixedCols` і `FixedRows` встановити рівними 0.

Відповідно до завдання встановіть граничні значення кількості рядків n і стовпців m для компоненту `StringGrid1`: `ColCount=8`, а `RowCount=6` (вісім стовпців і шість рядків). Для компоненту `StringGrid2` `ColCount=1`, `RowCount=8`, а для компоненту `StringGrid3` `ColCount=1`, `RowCount=6`.

За замовченням в компонент `StringGrid` заборонено введення інформації з клавіатури, тому для компоненту `StringGrid1` необхідно в Інспекторі Об'єктів двічі клацнути “мишею” на символі `+` властивості `+Options` і в списку опцій, що відкрився, встановити значення `goEditing` в `True`.

Для зручності роботи з компонентами `SpinEdit` встановити для компоненту **SpinEdit1** значення властивостей: `MinValue=1`, `MaxValue=6`, а для компоненту **SpinEdit2**: `MinValue=1`, `MaxValue=8`.

3. Створення процедур обробки подій `SpinEdit1Change` і `SpinEdit2Change`

Події `SpinEdit1Change` і `SpinEdit2Change` виникають при будь-якій зміні значення в полі редактора `SpinEdit1` і `SpinEdit2` відповідно. Створимо процедури обробки цих подій, в яких привласнимо значення n і m , одержемо поля редакторів `SpinEdit`, з властивостям `ColCount` і `RowCount` компонентів `StringGrid`. Це дозволить управляти розмірами таблиць `StringGrid` за допомогою компонентів `SpinEdit` без додаткових кнопок, оскільки зміна значень в полі редактора `SpinEdit` відразу приведе до зміни розміру таблиць `StringGrid`. Двічі клацніть “мишею” на компоненті `SpinEdit1` – курсор встановиться в тексті процедури-обробника події `SpinEdit1Change`:

```
procedure TForm1.SpinEdit1Change(Sender: TObject).
```

Уважно наберіть оператори цієї процедури, використовуючи текст модуля UnMas. Аналогічним чином створіть процедуру події SpinEdit2Change:

```
procedure TForm1.SpinEdit2Change(Sender: TObject).
```

4.Текст модуля UnMas:

```
Unit UnMas;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs
  StdCtrls, Spin, Grids;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    Label8: TLabel;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    StringGrid3: TStringGrid;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    SpinEdit3: TSpinEdit;
    SpinEdit4: TSpinEdit;
    Label6: TLabel;
    Label7: TLabel;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure SpinEdit2Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
```

```

Form1: TForm1;
implementation
{$R *.DFM}
var
  A:array[1..6,1..8] extended;// оголошення двовимірного масиву A
  X:array[1..8] extended; // оголошення одновимірного масиву X
  Y:array[1..6] extended; // оголошення одновимірного масиву Y
  n,m,p,q:integer; // оголошення глобальних змінних
procedure TForm1.FormCreate(Sender: TObject);
begin
  SpinEdit1.Text:='4'; // початкове значення n
  SpinEdit2.Text:='6'; // початкове значення m
  SpinEdit3.Text:='2'; // початкове значення p
  SpinEdit4.Text:='3'; // початкове значення q
  StringGrid1.RowCount:=4; // кількість рядків масиву A
  StringGrid1.ColCount:=6; // кількість стовпців масиву A
  StringGrid2.RowCount:=6; // кількість рядків масиву X
  StringGrid3.RowCount:=4; // кількість рядків масиву Y
end;
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
  n:=StrToInt(SpinEdit1.Text);// n привласнюється вміст поля редактора
  StringGrid1.RowCount:=n; // встановлюється кількість рядків масиву A
  StringGrid3.RowCount:=n; // встановлюється кількість рядків масиву Y
end;
procedure TForm1.SpinEdit2Change(Sender: TObject);
begin
  m:=StrToInt(SpinEdit2.Text);// m привласнюється вміст поля редактора
  StringGrid1.ColCount:=m; // встановлюється кількість стовпців масиву A
  StringGrid2.RowCount:=m; // встановлюється кількість рядків масиву X
end;
procedure TForm1.Button1Click(Sender: TObject);
var
  i,j:integer; // оголошення локальних змінних
begin
  n:=StrToInt(SpinEdit1.Text);
  StringGrid1.RowCount:=n;
  StringGrid3.RowCount:=n;
  m:=StrToInt(SpinEdit2.Text);
  StringGrid1.ColCount:=m;
  StringGrid2.RowCount:=m;
  p:=StrToInt(SpinEdit3.Text);
  q:=StrToInt(SpinEdit4.Text);
  // Введення значень з таблиці в масив A
  for i:=1 to n do

```

```

for j:=1 to m do
  A[i,j]:=StrToFloat(StringGrid1.Cells[j-1,i-1]);
for j:=1 to m do // формування масиву X і виведення його значень в таблицю
  begin
  X[j]:=A[p,j];
  StringGrid2.Cells[0,j-1]:=FloatToStrF(X[j],ffFixed,3,1);
  end;
for i:=1 to n do // формування масиву Y і виведення його значень в таблицю
  begin
  Y[i]:=A[i,q];
  StringGrid3.Cells[0,i-1]:=FloatToStrF(Y[i],ffFixed,3,1);
  end;
end;
end.

```

5. Робота з додатком.

Запустіть створений додаток. Занесіть числові значення в елементи матриці A і переконаєтеся в тому, що додаток функціонує відповідно до завдання.

Виконання індивідуального завдання

Запишіть в додаток 2 опис компонентів StringGrid і DrawGrid.

По вказівці викладача виберіть своє індивідуальне завдання. Створіть програму-додаток і протестуйте його роботу.

Індивідуальні завдання:

1. Задана цілочисельна матриця A розміром $N \times M$. Одержати масив B , привласнивши його k -му елементу значення 0 , якщо всі елементи k -го стовпця матриці нульові, і значення 1 інакше ($k=1, 2, \dots, M$).
2. Задана цілочисельна матриця A розміром $N \times M$. Одержати масив B , привласнивши його k -му елементу значення 1 , якщо елементи k -ї рядка матриці впорядковані по убуту, і значення 0 інакше ($k=1, 2, \dots, N$).
3. Задана цілочисельна матриця A розміром $N \times M$. Одержати масив B , привласнивши його k -му елементу значення 1 , якщо k -я рядок матриці симетричний, і значення 0 інакше ($k=1, 2, \dots, N$).
4. Задана цілочисельна матриця розміром $N \times M$. Визначити кількість “особливих” елементів матриці, вважаючи елемент “особливим”, якщо він більше суми решти елементів свого стовпця.
5. Задана цілочисельна матриця розміром $N \times M$. Визначити кількість “особливих” елементів матриці, вважаючи елемент “особливим”, якщо в

його рядку зліва від нього знаходяться елементи, менші його, а справа – великі.

6. Задана символна матриця розміром $N \times M$. Визначити кількість різних елементів матриці (тобто елементи, що повторюються, рахувати один раз).

7. Дана речовинна матриця розміром $N \times M$. Упорядкувати її рядки по неубуванню їх перших елементів.

8. Дана речовинна матриця розміром $N \times M$. Упорядкувати її рядки по неубуванню суми їх елементів.

9. Дана речовинна матриця розміром $N \times M$. Упорядкувати її рядки по неубуванню їх найбільших елементів.

10. Визначити чи є задана квадратна матриця n -го порядку симетричної відносно побічної діагоналі.

11. Для заданої цілої матриці розміром $N \times M$ вивести на екран всі її крапки. Елемент матриці називається крапкою, якщо він є як найменшим в своєму рядку і одночасно найбільшим в своєму стовпці або, навпаки, є найбільшим в своєму рядку і як найменшим в своєму стовпці.

12. У матриці n -го порядку переставити рядки так, щоб на головній діагоналі матриці були розташовані елементи, найбільшій по абсолютній величині.

Вимоги до оформлення звіту :

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Що таке компонент StrihgGrid?
2. Які має властивості компонент StrihgGrid?
3. Як встановити кількість рядків відповідного масиву?
4. Що таке фіксована зона?
5. Як створити процедуру обробки подій SpinEdit1Change і SpinEdit2Change?
6. Що таке компонент SpinEdit?
7. Які властивості компонента SpinEdit?
8. Як сформувати масив і вивести його значення в таблицю?

Практична робота №14.

Тема: Програмування алгоритмів з використанням динамічних структур даних.

Мета: освоїти методику створення програм-додатків, в яких використовуються динамічні структури даних.

Теоретичні відомості:

Приклади створення програм-додатків.

1. Використовування динамічних масивів.

Завдання 1:

Створити додаток для обчислення як найменшого і найбільшого зі всіх значень елементів цілочисельної матриці $A=\{i,j\}$, де $i=1,2., m$; $j=1,2., n$. Значення m і n задаються користувачем на панелі інтерфейсу, а елементи матриці A генеруються за допомогою датчика випадкових чисел і розміщуються в пам'яті динамічно.

Один з можливих варіантів панелі інтерфейсу створюваного додатку показаний на рис. 14.1.

2. Розміщення компонентів на Формі.

Розмістимо на Формі компоненти Label, SpinEdit, Button і StringGrid.

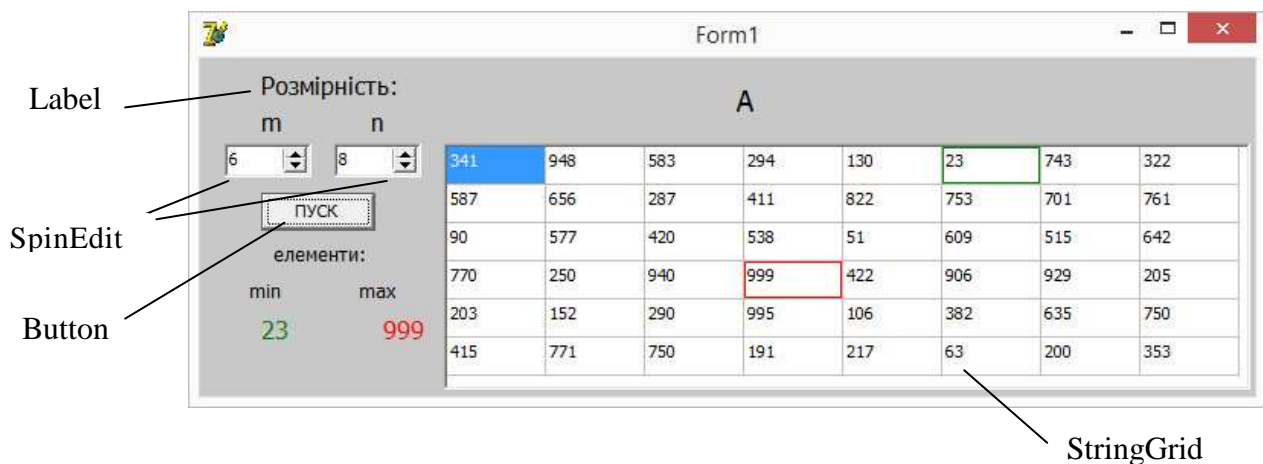


Рис. 14.1

Збережемо модуль під ім'ям UnDinMas (текст модуля приведений в п. 3).

3. Створення процедур обробки подій FormCreate і Button1Click.

Подвійним натисненням клавіші "миші" на Формі і кнопці Button1 створіть відповідні процедури обробки подій. Користуючись текстом модуля UnDinMas, уважно наберіть оператори цих процедур.

При бажанні можна створити процедуру, яка виділятиме заданим кольором межі осередків з як найменшим і найбільшим значеннями в компоненті StringGrid. Для створення такої процедури зробіть активним компонент StringGrid і на сторінці Events(події) Інспектора Об'єктів двічі клацніть “мишею” в правій частині події OnDrawCell. У відповідь Delphi створить обробник цієї події – процедуру **procedure TForm1.StringGrid1DrawCell** і встановить курсор між операторами **begin** і **end** цієї процедури. Використовуючи текст модуля UnDinMas, уважно наберіть оператори процедури TForm1.StringGrid1DrawCell.

4. Текст модуля UnDinMas:

```
Unit UnDinMas;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs
  StdCtrls, Spin, Grids, Buttons;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    Label8: TLabel;
    StringGrid1: TStringGrid;
    Label2: TLabel;
    Label5: TLabel;
    Label3: TLabel;
    Button1: TButton;
    Label4: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label9: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure SpinEdit2Change(Sender: TObject);
    procedure StringGrid1DrawCell(Sender: TObject; Col, Row: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```

var
  Form1: TForm1;

implementation
{$R *.DFM}
Type
Mas=array[1..1] integer;      // масив цілочисельних значень
pMas=array[1..1] ^mas;      // масив покажчиків
var                            // оголошення глобальних змінних
pA:^pMas;                    // покажчик на масив покажчиків
m,n,max,min:integer;
procedure TForm1.FormCreate(Sender: TObject);
begin
  m:=6;                       // початкове значення m
  n:=8;                       // початкове значення n
  SpinEdit1.Text:='6';
  SpinEdit2.Text:='8';
  StringGrid1.RowCount:=m;    // кількість рядків
  StringGrid1.ColCount:=n;    // кількість стовпців
end;
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
  m:=StrToInt(SpinEdit1.Text); // m привласнюється вміст поля редактора
  StringGrid1.RowCount:=m;
end;
procedure TForm1.SpinEdit2Change(Sender: TObject);
begin
  n:=StrToInt(SpinEdit2.Text); // n привласнюється вміст поля редактора
  StringGrid1.ColCount:=n;
end;
procedure TForm1.Button1Click(Sender: TObject);
label 1;
var
  i,j,k,l,r:integer;
begin
  Randomize;                  // ініціалізація датчика випадкових чисел
  GetMem(pA,4*m);            // виділення пам'яті для масиву з m покажчиків
  for i :=1 to m do
  begin                      // формування i-й рядка масиву
    { Виділення пам'яті для n елементів i-й рядка}
    GetMem(pA^[i],SizeOf(integer)*n);
    pA^[1]^1:=Random(1000);  // випадкове ціле число занести в масив
    for j:=1 to n do
    begin                    // формування j-го елементу рядка
      1: r:=Random(1000);    // генерація випадкового числа
    end;
  end;
end;

```

```

for до:=1 to i do
  for l:=1 to j do
    if r=pA^[k]^[l] then // якщо таке число вже є в масиві тоді...
      goto 1;
    pA^[i]^[j]:=r; // випадкове число занести в масив
  end;
end;
for i:=1 to m do // елементи масиву занести в осередки
  for j:=1 to n do // компоненти StringGrid1
    StringGrid1.Cells[j-1,i-1]:=IntToStr(pA^[i]^[j]);
  { Пошук min і max значень серед елементів масиву}
  max:=pA^[1]^[1];
  min:=max;
  for i:=1 to m do
    for j:=1 to n do
      if pA^[i]^[j]<min then
        min:=pA^[i]^[j]
      else
        if pA^[i]^[j]>max then
          max:=pA^[i]^[j];
Label7.Caption:=IntToStr(min); // висновок min значення
Label9.Caption:=IntToStr(max); // висновок max значення
for i:=1 to m do
  { Звільняє пам'яті, займаної n елементами i-й рядка}
  FreeMem(pA^[i],SizeOf(integer)*n);
  { Звільнення пам'яті, займаної масивом з t покажчиків}
  FreeMem(pA,4*m);
end;
procedure TForm1.StringGrid1DrawCell(Sender: TObject; Col, Row: Integer;
  Rect: TRect; State: TGridDrawState);
begin
  with StringGrid1.Canvas do
    if StringGrid1.Cells[Col,Row]=IntToStr(min) then // якщо елемент осередку
      begin // рівний min тоді...
        Brush.Color:=clGreen; // встановити колір зелений
        FrameRect(Rect); // виділити межі осередку заданим кольором
      end
    else
      if StringGrid1.Cells[Col,Row]=IntToStr(max) then // якщо елемент осередку
        begin // рівний max тоді...
          Brush.Color:=clRed; // встановити колір кисті червоний
          FrameRect(Rect); // виділити межі осередку заданим кольором
        end
      end;
end.
end.

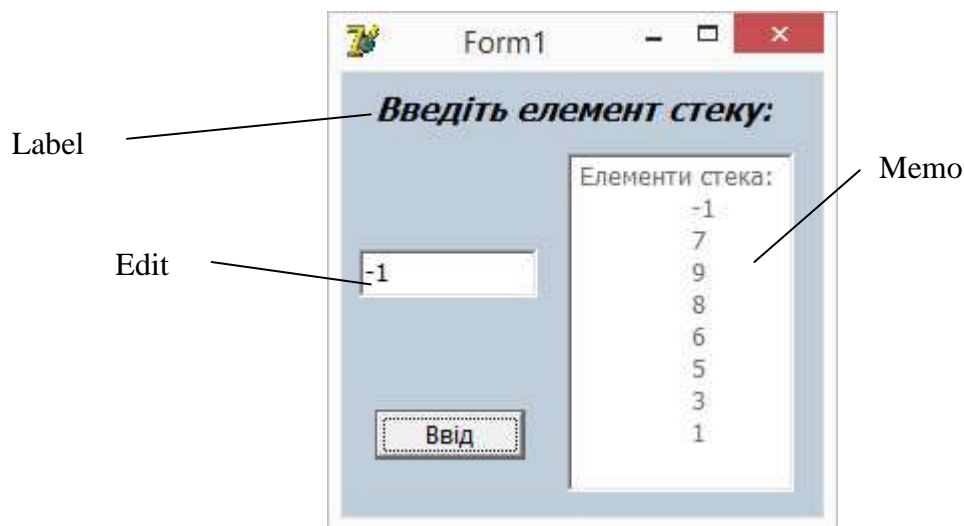
```

5. Використання динамічних списків.

Завдання2:

Створити програму-додаток для формування стека, який заповнюється шляхом введення цілих позитивних чисел з клавіатури. Як тільки буде введено перше негативне число, вміст стека виводиться на панель інтерфейсу, а пам'ять займана його елементами звільняється.

Один з можливих варіантів панелі інтерфейсу створюваного програмного- додатку показаний на рис. 15.2.



6. Розміщення компонентів на Формі.

Розмістимо на Формі компоненти Label, Edit, Button і Мемо(Рис. 15.2). Збережемо модуль під ім'ям UnStek.

7. Створення процедур обробки подій FormCreate і Button1Click.

Подвійним натисненням клавіші "миші" на формі і кнопці Button1 створіть відповідні процедури-обробки подій. Використовуючи текст модуля UnStek, уважно наберіть оператори цих процедур.

8. Текст модуля UnStek:

```
Unit UnStek;  
interface  
uses  
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs  
  StdCtrls;  
type  
  TForm1 = class(TForm)  
    Label1: TLabel;  
    Edit1: TEdit;  
    Button1: TButton;
```

```

Label2: TLabel;
Label3: TLabel;
Memo1: TMemo;
procedure Button1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
Type
PSt:=^Zap;
Zap=record
  inf:integer;
  adr:PSt
end;
Var      // оголошення глобальних змінних:
PVer,    // покажчик вершини стека
PTek:PSt; // поточний покажчик
ElSt:integer; // елемент стека
procedure TForm1.Button1Click(Sender: TObject);
begin
  New(PTek);          // виділити пам'ять
  ElSt:=StrToInt(Edit1.Text); // в ElSt занести значення з Edit1
  PTek^.inf:=ElSt;   // в інформаційну частину стека занести ElSt
  PTek^.adr:=PVer;   // в адресну частину занести покажчик на вершину
  PVer:=PTek; // покажчик вершини повинен указувати на останній елемент
  if ElSt>=0 then    // якщо елемент стека ненегативний тоді...
    begin
      Edit1.Text:="";          // очистити вікно редактора Edit1
      Edit1.SetFocus;         // передати фокус введення редактору Edit1
    end
  else
    begin
      Memo1.Lines.Add('Елементи стека:'); // вивести заголовок
      repeat
        Memo1.Lines.Add(#9+IntToStr(PTek^.inf)); // виведення елементів
        PVer:=PTek^.adr;
        Dispose(PTek); // звільнити пам'ять
        PTek:=PVer
      until PTek=nil;
    end
  end;

```

```

        end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
    PVer:=nil;           // ініціалізувати покажчик вершини
    ElSt:=0;            // ініціалізувати елемент стека
end;
end.

```

9. Індивідуальне завдання:

По вказівці викладача виберіть два варіанти індивідуальних завдань. В завданнях №1-№15 необхідно використовувати динамічні масиви, а в завданнях №16-№30 – динамічні списки. У всіх завданнях необхідно передбачити контрольний висновок початкових даних.

1. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури, міняє місцями елементи з найбільшим і якнайменшим значеннями серед парних і виводить одержаний масив.
2. Створити програму-додаток, яка здійснює введення m рядків і n стовпців двовимірного масиву з клавіатури і виводить номер рядка і номер стовпця якнайменшого зі всіх значень його елементів.
3. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури і виводить порядковий номер елемента з якнайменшим значенням серед непарних.
4. Створити програму-додаток, яка здійснює введення значень елементів двовимірного масиву n -го порядку з клавіатури і виводить значення найбільшого з елементів головної діагоналі.
5. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури, змінює порядок проходження елементів на протилежний і виводить одержаний масив.
6. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури і виводить порядковий номер елемента з найбільшим значенням серед парних.
7. Створити програму-додаток, яка здійснює введення значень елементів двовимірного масиву n -го порядку з клавіатури і виводить значення суми елементів головної діагоналі.

8. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури, міняє місцями елементи з мінімальним і максимальним значеннями і виводить одержаний масив.
9. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури і виводить порядковий номер елемента з якнайменшим значенням серед позитивних.
10. Створити програму-додаток, яка здійснює введення значень елементів двовимірного масиву n -го порядку з клавіатури, змінює порядок проходження елементів головної діагоналі на протилежний і виводить перетворений масив.
11. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури, міняє місцями елементи з мінімальним і максимальним значеннями серед позитивних і виводить одержаний масив.
12. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури і виводить порядковий номер елемента з найбільшим значенням серед негативних.
13. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури, міняє місцями елементи з найбільшим значенням серед негативних і якнайменшим серед позитивних і виводить одержаний масив.
14. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури і виводить середнє арифметичне значення елементів масиву.
15. Створити програму-додаток, яка здійснює введення до значень елементів одновимірного масиву з клавіатури, міняє місцями елементи з якнайменшим значенням серед парних і найбільшим серед непарних і виводить одержаний масив.
16. Створити програму-додаток, яка заносить в стек цілі позитивні числа з клавіатури, виводить вміст стека і середнє арифметичне значення його елементів.
17. Створити програму-додаток, яка заносить в стек символи з клавіатури, виводить вміст стека і повідомлення про те, міститься чи ні в стеку заданий символ.

18. Створити програму-додаток, яка заносить в кожний елемент стека англійське слово з клавіатури і, як тільки буде введено слово “end”, виводить вміст стека.
19. Створити програму-додаток, яка заносить в стек довільні цілі числа з клавіатури, виводить вміст стека і повідомлення про те, міститься чи ні в стеку задане число.
20. Створити програму-додаток, яка заносить в стек символи з клавіатури, виводить вміст стека і повідомлення про те, чи впорядковані елементи стека за абеткою чи ні.
21. Створити програму-додаток, яка заносить в стек позитивні цілі числа з клавіатури і, як тільки буде введено число, рівне сумі введених чисел, виводить вміст стека.

Вимоги до оформлення звіту :

Звіт повинен містити:

1. Тему та мету лабораторної роботи.
2. Завдання на лабораторну роботу.
3. Текст програми.
4. Відповідь на контрольні запитання.

Контрольні запитання:

1. Як оголосити масив цілочисельних значень?
2. Як в інформаційну частину стека занести ElSt?
3. Що таке стек.

Практична робота № 15.

Тема: Робота з графікою.

Мета: Навчитись використовувати графічні функції, при написанні програм.

Теоретичні відомості:

1. Загальні положення

Можливості графіки в системі Delphi реалізовані через так званий дескриптор контексту графічного пристрою DC і три його інструменти – шрифт, перо, пензель. Відповідно до цього в Delphi створені класи, що дають змогу використовувати графічні інструменти Windows: для контексту – клас TCanvas, для шрифту – TFont, для пера – TPen, для пензля – TBrush. Пов'язані з цими класами об'єкти автоматично створюються для всіх видимих елементів і стають доступними через властивості **Canvas**, **Font**, **Pen**, **Brush**.

Canvas (канва, полотно) – це площа видимого елемента, на якій можна малювати. Вона є однією з властивостей видимого елемента, **Font**, **Pen** та **Brush** є підвластивостями в **Canvas**.

Pen (перо) – це лінія, яка виділяє зовнішній контур фігури. Можна задавати товщину, колір, вигляд (тип) такої лінії.

Brush (пензель) – це площа фігури, тобто внутрішня від контура частина. Можна задавати колір площини фігури та спосіб зафарбовування, наприклад, горизонтальними чи діагональними лініями.

Font – використовують для вибирання шрифтів, які застосовують під час виведення тексту за допомогою методів TextOut або TextRect об'єкта **Canvas**.

У процесі розробки проекту властивість Canvas недоступна у вікні Object Inspector. Щоб використати Canvas, треба писати оператори, які присвоюють властивостям Canvas необхідні значення та викликають методи Canvas. Наприклад, щоб намалювати на формі голубий прямокутник, заповнений жовтими діагональними штрихами, треба ввести в опрацювання події форми OnPaint такі оператори:

```
With Canvas do {малюватимемо на полотні форми}
begin
  Pen.Color:=clBlue; {колір ліній – Голубий}
  Brush.Color:=clYellow; {колір зафарбовування – жовтий}
  Brush.Style:=bsDiagCross; {спосіб зафарбовування – діагональні лінії}
  Rectangle(10,10,100,100);
end;
```

Не всі компоненти мають властивість Canvas, наприклад, її не мають панелі – компоненти типу TPanel. Однак і на таких компонентах можна малювати, розмістивши на них спеціальну компоненту типу TPaintBox, яка власне і призначена для малювання. Крім того, об'єкти PaintBox можна використовувати для обмеження на формі чи на деякій компоненті ділянки малювання одним чи декількома прямокутниками. Коротка характеристика властивостей та методів Клас TPen:

Color – колір лінії;

Width – товщина лінії в пікселях;

Style – стиль (зображення) лінії;

Mode – метод малювання лінії, тобто спосіб накладання кольору лінії на поверхню. Клас TBrush:

Color – колір зафарбовування;

Style – спосіб зафарбовування внутрішньої площини фігури. Клас TFont;

Name – назва шрифту, наприклад, Arial;

Style – стиль шрифту: грубий, курсив, підкреслений, перекреслений;

Color – колір шрифту; *Size* – розмір шрифту в пунктах. Клас TCanvas:

Pen, Brush, Font – визначено вище;

Pixels – двовимірний масив точок для доступу до окремих пікселів Об'єкта Canvas.

Крім того, є низка методів для малювання різноманітних фігур: *Arc, Chord, Ellipse, LineTo, FillRect, Draw, Polygon, Rectangle* тощо.

Приклад 1. Вивчити можливості малювання ліній та фігур у графічній підсистемі Delphi – стилі та кольори ліній, кольори та способи зафарбовування площини фігур. Малювати на формі, яка не містить жодних компонент, у процедурі опрацювання події форми *OnPaint*.

Нижче наведено приклад реалізації цього завдання. Варто звернути увагу на спосіб, за яким на форму можна виводити готові Малюнки – фотографії, схеми тощо, підготовлені попередньо в *bmp*-файлах.

```

unit Unit1;
interface uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1= class(TForm)
    procedure FormPaint(Sender: TObject); procedure
    FormCreate(Sender: TObject); procedure FormDestroy(Sender:
    TObject);
  private (Private declarations)
    gr: TBitmap; {для запам'ятовування малюнка з файлу}
  public
    {Public declarations} end;
  var
    Form1: TForm1;
  implementation
    {$R*.DFM}
  procedure TForm1.FormPaint(Sender: TObject) ;
    {метод виконується у випадку, кожного перемальовування
    форми}
    var
      a: integer; R: TRect;
    begin
      With Canvas do :
      begin
        Pen.Color:=clBlue; {спочатку визначаємо кольори та стиль зафарбовування}
        Brush.Color:=clYellow; Brush.Style:=bsDiagCross; ,
        for a:=1 to 5 do {малюємо п'ять прямокутників, визначаючи їхнє
        розташування та розміри}
          Rectangle((a-1)*100+10,25,a*100,115); Brush.Style:=bsSolid; {тепер
          змінюємо стиль, не змінюючи кольорів}
          for a:=1 to 5 do {малюємо п'ять еліпсів}
            Ellipse((a-1)*100+10,140,a*100,200); {вивести в нижній частині форми готовий
            малюнок}
            .R.Left:=30; R.Right:=370;
            R.Top:=220; R.Bottom:=450;
            StretchDraw(R,gr); {цей метод переносить малюнок у межі
            заданого прямокутника R}
            end;
            end;
      end;
    procedure TForm1.FormCreate(Sender: TObject); {метод виконується один раз під
    час першого створення форми}
    begin
      gr:=TBitmap.Create; (створюємо об'єкт для картинки) //і завантажуюмо файл
      з картинкою з поточного каталогу

```

```

gr.LoadFromFile('Univerl.bmp'); {для завантаження є готовий метод}
end;
procedure TForm1.FormDestroy(Sender: TObject) ;
(метод виконується один раз під час остаточного закриття форми –
закінчення роботи)
begin
gr.Free; //вивільнити пам'ять у разі закінчення роботи
end;
end.

```

Приклад 2 (для самостійного виконання). Придумати та вивести на форму складніший малюнок, який би складався з геометричних фігур. Фігури зафарбовувати різними кольорами та способами. Для цього треба переглянути довідкову систему Delphi і відшукати там перелік назв кольорів, стилів ліній, способів зафарбовування, а також перелік методів Canvas для безпосереднього малювання.

Приклад 3 (для самостійного виконання). Підібрати три готові *bmp*-файли. Розташувати на формі шість кнопок – по дві на кожен *bmp* – файл. У разі натискання на одну кнопку картинка з *bmp* – файлу повинна виводитися на форму, а внаслідок натискання на іншу – стиратися. Картинки повинні виводитися незалежно одна від одної.

Приклад 4 (для самостійного виконання). Розділивши умовно форму приблизно навпіл, розташувати на ній дві компоненти типу *TPaintBox*. Малювати не на формі безпосередньо, а на об'єктах *PaintBox*. На кожному з двох таких об'єктів виконувати малювання після клацання на цьому об'єкті мишкою (подія *OnClick*). У разі подвійного клацання на об'єкті (подія *OnDblClick*) малюнок повинен стиратися.

2. Побудова та перегляд графіків функції

Графіки функцій можна будувати та переглядати різними способами. Нижче описано один із способів – пряме виведення на панель малювання на формі графіка наперед заданої функції однієї змінної без збереження результатів обчислень у файлі. Однак у діалозі можна визначати ліву та праву межі зміни значень аргументу, а також максимальне за модулем значення функції. Крім того, демонструються деякі корисні прийоми роботи з компонентами, розташованими на формі.

Приклад 1. Створити форму для переглядання графіків, як показано на рис. 15.1

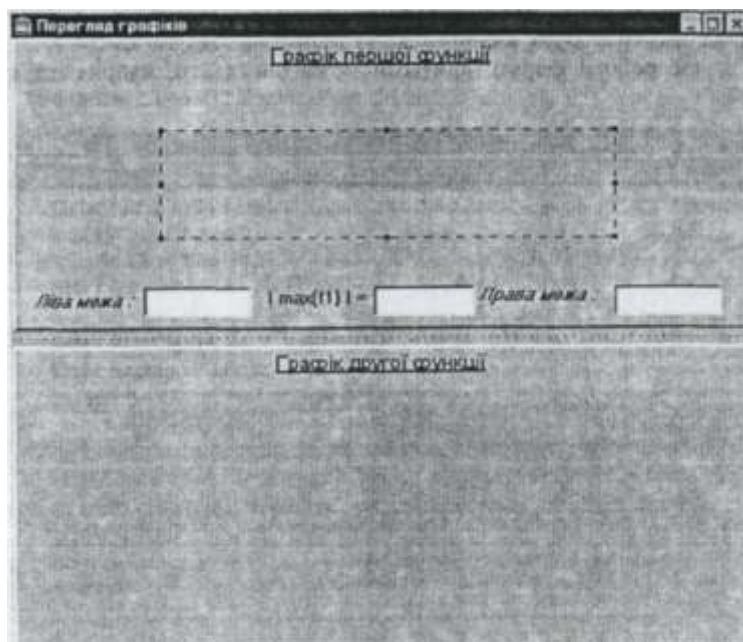


Рис. 15.1 – Проект форми для переглядання графіків

На формі розміщені дві панелі. Верхня вирівняна до верхньої межі форми, а нижня – до нижньої. Висоту панелей можна задати під час проектування або програмне під час виконання. На малюнку для прикладу спроектована лише верхня панель. Нижня – ідентична.

На кожній панелі повинно бути: зверху – підпис, посередині – панель для малювання *PaintBox*, знизу – три поля редакторів для введення чисел і відповідні підписи біля цих полів. У наведеному нижче роздруку тексту програми зверніть увагу на те, як вирівнюються елементи на панелі у разі зміни її розмірів. Отже, під час проектування форми не є дуже важливим місце розташування елементів, однак їхні розміри зафіксовані і не змінюються під час виконання програми, хоча і їх також можна було б змінювати.

Сам графік функції з'являється на панелі *PaintBox* внаслідок подвійного клацання на ній мишкою.

Ліву та праву межі вводять як дійсні числа у прийнятому в Delphi форматі. Вони означають інтервал за X , на якому відображається графік функції. Зазначимо, що, вводячи дробову частину числа, її треба відділяти *комою*, а не крапкою. Те саме стосується максимального за модулем видимого значення функції. Якщо фактичне значення функції його перевищує? то графіка просто не буде видно.

Під час роботи форма програми може виглядати, наприклад, як на рис 15.2.

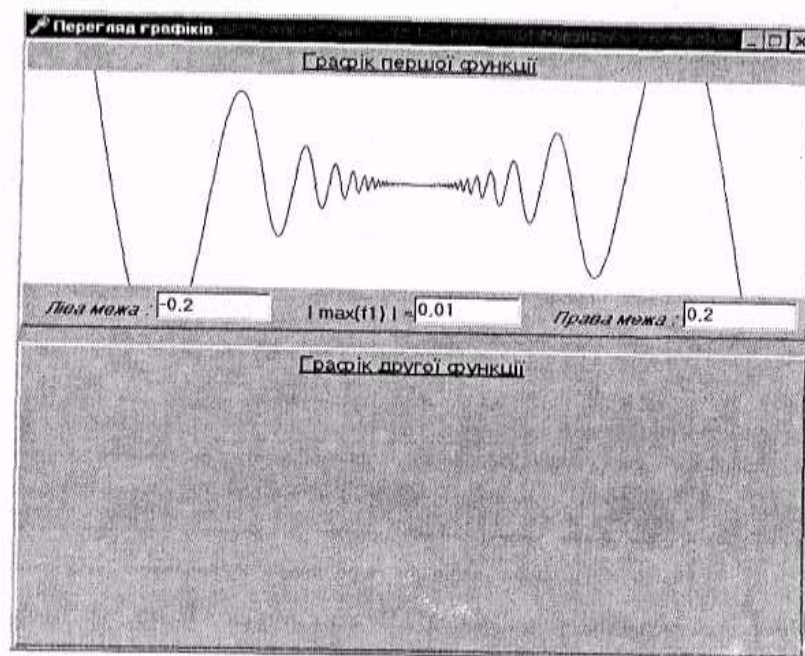


Рис. 15.2. – Виконання програми переглядання графіків.

Наведений роздрук відображає лише половину завдання – малює графік тільки першої функції на верхній панелі. Доповніть форму і програму так, щоб можна було малювати і другий графік на нижній панелі. Це виконують майже ідентично, змінюються лише імена компонент та імена опрацювання їхніх подій.

Працюючи з готовою програмою, зверніть увагу на те, що у разі зміни розмірів вікна графік зникає з панелі. Те ж саме відбувається у випадку мінімізації вікна та його наступного відновлення. Отже, після зміни стану вікна треба знову двічі клацнути мишкою на панелі малювання для повторного відображення графіка.

Приклад реалізації завдання:

```

unit Unit1;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, ExtCtrls;
type
TForm1 = class(TForm)
Panel1: TPanel;
Panel2: TPanel;
LabelFn1: TLabel;
LabelFn2: TLabel;
LeftFl: TLabel;
RightFl: TLabel;
EditL1: TEdit;

```



```

EditR1: TEdit;
Lmax1: TLabel;
EditMax1: TEdit;
PaintBox1: TPaintBox;
procedure PanellResize(Sender: TObject);
procedure EditL1Exit(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure EditR1Exit(Sender: TObject);
procedure EditMax1Exit(Sender: TObject);
procedure PaintBox1Db1Click(Sender: TObject);
private
{Private declarations }
public
{Public declarations }
afl, bfl, af2, bf 2: real ; //ліві та праві межі
//інтервалів
AbsMaxF1, AbsMaxF2: real; //максимальні за модулем
//значення
rdaf1, rdbfl, rdaf2, rdbf2: boolean; //ознаки введення
rdAbsMaxF1, rdAbsMaxF2: boolean;
end;
var
Form1: TForm1;
implementation
{$R*.DFM} .
//сталі для повідомлень у випадку помилок під час введення чисел
const
err = 'помилка!';
txtl = 'Неправильно задана ліва межа' ;
txtr = 'Неправильно задана ліва межа' ;

```

Модуль програми:

```

txtm = 'Неправильно задане максимальне значення'; //опишемо функції,
графіки яких будуватимемо Function F1(x: real): real; begin try
Result:= x*x*Sin(1/x) except
on EZeroDivide do Result:=0 end; end;
Function F2(x: real): real; begin
Result:=Sin(x/Ln(Abs(x)+1.1)); end;
procedure TForm1.FormCreate(Sender: TObject); {у разі створення форми
присвоюємо початкові значення}
begin
// спочатку значення ще не введені
rdaf1:=false; rdbf 1: =false,- rdaf2:=false; rdbf2:=false; rdAbsMaxF1:=false;
rdAbsMaxF2:=false; end;

```

```

procedure TForm1.PanellResize(Sender: TObject); //ця процедура виконується у
випадку будь-якої зміни
    //розмірів першої (верхньої) панелі begin
    //Усі елементи в нижній частині панелі вирівнюються до її
        //нижньої межі //однаково за відкиданням висоти і ще 10 (властивість
        //Top)//Крім того, елементи вирівнюються за шириною (Left),
        //однак по-різному
    //підпис Ліва межа вирівнюється до лівого боку панелі With LeftFl do begin
        Left:=20; Top:=Panel1.ClientHeight-Height-10; end; //підпис Права межа повинен
бути лівіше від поля введення
        //такого числа With RightFl do begin
Left:=Panel1.ClientWidth-Width-EditRl.Width-10;      Top:=Panel1.ClientHeight-
Height-10 ; end; //поле редактора для введення лівої межі повинне бути
        //правіше від підпису With EditLl do begin
        Left:=20+LeftFl.Width;
        Top:=Panell.ClientHeight-Height-10; end; //поле редактора для введення правої
межі вирівнюється до
        //правого боку панелі With EditRl do begin
        Left:=Panell.ClientWidth-Width-10;
        Top:=Panell.ClientHeight-Height-10; end;
    //підпис "max(fl)=" вирівнюється до середини With Lmaxl do begin
        Left:=(Panel1.ClientWidth-Width-EditMaxl.Width)div2 ;
        Top:=Panell.ClientHeight-Height-10; end; //поле редактора для введення
максимального значення
        //також вирівнюється до середини With EditMaxl do begin
        Left:=(Panel1.ClientWidth-Width+Lmaxl.Width)div2;
        Top:=Panell.ClientHeight-Height-10; end;
    //панель для малювання PaintBox1 вирівнюється до середини With PaintBox1 do
begin
        Left:=3; Top:=LabelFnl.Height+5;
        Width:=Panell.ClientWidth-6;
        Height:=EditLl.Top-Top-5;
end;
procedure TForm1.EditLlExit(Sender: TObject);
(метод виконується у разі втрати фокуса редактором EditLl •
введення лівої межі} begin try
    afl:=StrToFloat(EditLl.Text); rdafl:=true; (якщо ввели правильне
значення} except
    on EConvertError do (неправильний формат числа} begin rdafl:=false;
        Application.MessageBox(txtl, err,
        mb_IconExclamation+mb_Ok); end;end;end;
procedure TForm1.EditRlExit(Sender: TObject); (метод виконується у разі втрати
фокуса редактором EditRl
введення правої межі}
begin

```

```

    tru
    bfl:=StrToFloat(EditRl.Text);   rdbfl:=true;   (якщо ввели правильне
значення} except
    on EConvertError do (неправильний формат числа} begin rdbfl:=false;
        Application.MessageBox(txtr, err,
        mb_IconExclamation+mb_Ok); end; end; end;
procedure TForm1.EditMaxlExit(Sender: TObject); (метод виконується у разі втрати
фокуса редактором EditMaxl
- максимального значення} begin try
AbsMaxFl:=StrToFloat(EditMaxl.Text);   rdAbsMaxFl:=true;   (якщо ввели
правильне значення} except
    on EConvertError do (неправильний формат числа} begin
        rdAbsMaxFl:=false;
        Application.MessageBox(txtm, err,
        mb_IconExclamation+mb_Ok); end; end; end;
procedure TForm1.PaintBox1Db1Click(Sender: TObject); var
    mky, mkx: real; //коефіцієнти для переобчислення точок
        //графіка
    x, y: real; //поточні значення
    px, py: integer; //значення, переобчислені в пікселі
    i: integer; begin
//Подвійним клацанням мишкою на панелі малюємо графік
//функції //спочатку примусово переміщуємо фокус уведення, щоб
//викликати подію OnExit для останнього введеного //значення Panel1. Set Focus;
//для панелі Panel 1 задати TabStop-м
//в інспекторі об'єктів if notfrdafl and rdbfl and rdAbsMaxFl) then Exit; //не
//все задано
//Графік повинен поміститися на панелі PaintBox1 //спочатку очистимо панель
PaintBox1, де буде графік
with PaintBox1 do begin
    Canvas.Brush.Color:=clWhite; Canvas.FillRect(ClientRect); end; //масштабний
коефіцієнт за Y - кількість пікселів на
//одиницю значення функції
mky:^PaintBox1.ClientHeight/(2*Abs(AbsMaxFl)); //по горизонталі розтягнемо
графік на ширину PaintBox1, //залишивши на краях по 5 пікселів //крок зміни
функції за X на 1 піксель: mkx:=(bfl-afl)/(PaintBox1.ClientWidth-10); if mkx<0 then
mkx:=Abs(mkx); (якщо переплутали межі} x:=afl; y:=Fl(x);
//координати в точках площини (пікселях): px:=5; py:=PaintBox1.ClientHeight
div2-round(y*mky) ; //перша точка графіка: PaintBox1.Canvas.MoveTo(px, py);
//графік у циклі: for i:=1 to PaintBox1.ClientWidth-10 do (через кожен
піксель за X} begin
    x:=afl-t-i*mkx; y:=Fl(x);
px:=5+i; py:=PaintBox1.ClientHeight div2-round(y*mky);
PaintBox1.Canvas.LineTo(px, py); end; end; end.

```

Приклад 2 (для самостійного виконання). Додайте до графіка функції, який виводиться, зображення осей координат (бажано іншим кольором). По краях осей виведіть підписи зі значеннями x_{min} , x_{max} , y_{min} , y_{max} , у градусах. Спробуйте по осі X вивести позначки (10-15 штук), також підписані відповідними значеннями аргументу.

Приклад 3 (для самостійного виконання). Виконати автоматичне масштабування значень функції (по осі Y) так, щоб завжди були видимими найбільші та найменші значення у випадку побудови графіка. Для цього можна забрати з форми підпис та поле редактора для введення модуля максимального значення. Запропонуйте один чи декілька методів обчислення найбільшого значення функції.

Приклад 4 (для самостійного виконання). Реалізуйте можливість перегляду графіків різних функцій із заздалегідь складеного переліку. Для цього доповніть панель компонентою TListBox, за допомогою якої можна вибирати погрібну функцію. Графіки різних функцій виводу під різними кольорами на ту саму панель, тобто графіки будуть накладатися. Перебачимо можливість очищення поля, на якому зображені графіки.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Юрченко І. В. Інформатика та програмування. Частина 1. Навчальний посібник. / І. В. Юрченко. – Чернівці : Книги–ХХІ, 2011. – 203 с.
2. Юрченко І. В. Інформатика та програмування. Частина 2. / І. В. Юрченко, В. С. Сікора. – Чернівці : Видавець Яворський С.Н., 2015. – 210 с.
3. Семенюк А. Д. Програмування: практикум / А. Д. Семенюк, Ф. О. Сопронюк. – Чернівці : Рута, 2010. – 143 с.
4. Фаронов В. В. Delphi. Программирование на языке высокого уровня / В. В. Фаронов. – СПб. : Питер, 2012.– 640 с.
5. Гофман В. Э. Delphi 7 / В. Э. Гофман, А. Д. Хомоненко. – М. : Питер, 2008. – 1145 с.
6. Культин Н. Б. Delphi 7. Программирование на Object Pascal / Н. Б. Культин. – М.: Питер, 2009.– 526 с.
7. Симонович С. Практическая информатика: универсальный курс / С. Симонович, Г. Евсеев. – М. : Инфорком–Пресс, 2009. – 480 с.
8. Руденко В. Д. Базовий курс інформатики / за заг. ред. В.Ю.Бикова. – К. : ВНУ, 2005 – Кн. 1: Основи інформатики. – 320 с.
9. Руденко В. Д. Базовий курс інформатики / за заг. ред. В.Ю.Бикова. – К. : Вид. група ВНУ, 2011 – Кн. 2: Інформаційні технології. – 368 с.

Навчальне видання

Інформатика

Методичні рекомендації

Укладач: **Волосюк** Юрій Вікторович

Формат 60x84 1/16 Ум. друк. арк. 3,8.

Тираж 30 прим. Зам. № _____

Надруковано у видавничому відділі
Миколаївського національного аграрного університету.
54020 м. Миколаїв, вул. Паризької комуни, 9

Свідоцтво суб'єкта видавничої справи ДК № 4490 від 20.02.2013 р.