

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

ІНСТРУКТИВНО-МЕТОДИЧНІ МАТЕРІАЛИ

для виконання лабораторних робіт

з дисципліни

«Інструментальні засоби візуального програмування»

для вищих навчальних закладів 1 та 2 рівня
акредитації

зі спеціальності 121

«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Розробила викладач _____ Н.В.Гапоненко

Розглянуто та затверджено комісією
програмної інженерії

Протокол № _____ від _____ . 20 ____ р.

Голова ЦК ПІ: _____ С.С. Ланська

Дніпро,
2023

ЛАБОРАТОРНА РОБОТА № 1-2

Створення проєкту в середовищі візуального програмування. Використання компонентів «надпис», «однорядковий редактор», «кнопка». Обробка події кліку по об'єкту

Мета: вивчити розміщення інструментарію середовища візуального програмування та структуру проєкту, навчитись використовувати компоненти «надпис», «однорядковий редактор», «кнопка», обробляти події кліку по об'єкту

Порядок проведення роботи

1 Ознайомитись з основними елементами середовища візуального програмування на прикладі C++ Builder.

Основними елементами середовища візуального програмування є:

- головне вікно з головним меню, палітрою інструментів та палітрою компонентів;
- текстовий редактор (Code Editor), відображуваний одним або декількома вікнами;
- конструктор форм (Form), відображуваний одним або декількома вікнами - по одному на кожну форму;
- інспектор об'єктів (Object Inspector).

2 Деякі компоненти можуть діяти подібно контейнерам компонентів (компоненти-батьки). А компоненти, призначені для змісту в контейнерах, називаються дочірніми. Прикладом таких взаємин може служити компонент TGroupBox. При перетаскуванні групи в інше місце кнопка переміщається разом з нею. Якщо помістити, наприклад, кнопку поза компонентом TGroupBox, то вона стане дочірнім об'єктом форми (для переміщення треба "вирізати" і вставити). Такі властивості позиціювання, як Left і Top, розглядаються щодо координат батьківського компонента.

3 Групу компонентів можна вибрати шляхом перетаскування прямокутника вибору за допомогою миші або шляхом натискання й утримання натиснутої клавіші <Shift> при виборі декількох компонентів.

Завдання

Пояснити у звіті призначення основних елементів середовища візуального програмування. Пояснити кроки зі створення проєкту. Дати заголовок формі з власним прізвищем. Розмістити на формі компоненти TGroupBox, в ньому «надпис», «однорядковий редактор», «кнопка». Пояснити, як декількома способами виділити одночасно «надпис» і «однорядковий редактор», виділіть їх і одночасно задайте жирний шрифт.

4. Вирівнювання компонентів

Для вирівнювання компонентів на формі можна використати наступні комбінації клавіш:

Shift + стрілки Змінює розмір компонента на один піксел у напрямку обраної стрілки.

Shift + Ctrl + стрілки Переміщає компонент на одну одиницю сітки в напрямку обраної стрілки.

Ctrl + стрілки Переміщає компонент на один піксел у напрямку обраної стрілки

Компоненти групи можна вирівнювати за розміром та взаємному розташуванню, як наведено нижче.

1. Виділити компоненти Edit, Memo і Button, вибрати правою клавішею миші команду Position|Align. У вікні, що відкрилося, “Вирівнювання розміщення” ліва частина Horizontal установлює вирівнювання компонентів по горизонталі. Тут

No change - не змінювати;

Left Sides - вирівняти по лівих сторонах;

Center - вирівняти по центрах;

Right Sides - вирівняти по правих сторонах;

Space equelly - розмістити з рівним інтервалом між координатами;

Center in Window - у центрі вікна.

Права частина вікна Vertical установлює вирівнювання компонентів по вертикалі.

Вибрати по черзі кожний з варіантів, описати один з результатів роботи.

2. Виділити компоненти Edit, Memo і Button, вибрати команду Position|Size. У вікні, що відкрилося, “Вирівнювання розмірів” ліва частина Width установлює вирівнювання компонентів по ширині. Тут

No change - не змінювати;

Shrink to Smallest - зменшити до розміру мінімального з компонентів групи;

Grow to largest - збільшити до розміру максимального з компонентів групи;
Width - задати у вікні ширину компонента в пікселях;

Можна змінити умови вирівнювання компонент, використовуючи пункт меню Tools/Options/User Interface. У групі Form designer вибрані наступні опції:

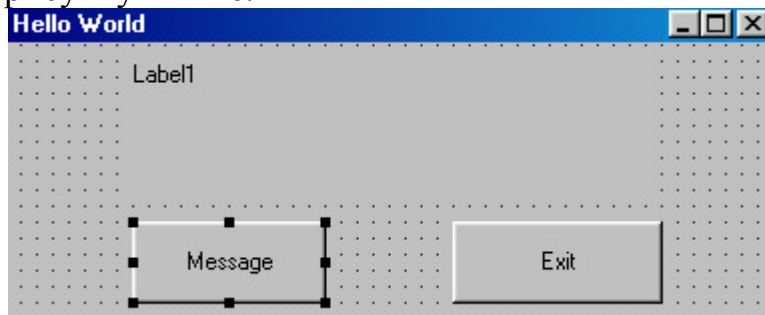
Display grid - зробити сітку із крапок на формі видимою для компонентів, що вирівнюються

Snap to grid - змусити ліві й верхні сторони компонентів розташуватися на лініях сітки.

Робота з проектом.

У створеному візуальному додатку виконати такі дії:

- Виберіть форму, інспектор об'єктів буде показувати її властивості. Виберіть у лівому стовпчику сторінки Properties властивість Caption. Замініть його значення (у правому стовпчику) на "Hello World". Цей текст відразу ж з'являється в рядку заголовка форми.
- На формі потрібні компоненти - дві кнопки Button1, Button2, мітка Label1, у якій буде відображатися необхідний текст.
- В інспекторі об'єктів для Button1 змініть властивість Caption кнопки на "Message", аналогічно тому, як це було зроблено для самої форми. Уведений текст задає напис на кнопці. (за замовчуванням це Button1.)
- Розмістіть на формі другу кнопку з написом "Exit", що зображено на рисунку нижче.



- Для Label1 в інспекторі об'єктів задайте властивість Font. Якщо нажати маленьку кнопку із багатокрапками у правому стовпчику інспектора, з'явиться звичайний діалог вибору шрифту, у якому можна задати бажану гарнітуру й розмір тексту. Кнопка із багатокрапками у колонці значень - звичайна ознака того, що для властивості передбачений спеціалізований редактор.
- Видаліть із властивості Caption весь текст ("Label1").

До даного моменту практично завершується те, що називають етапом візуального проектування програми. Тепер потрібно написати програмний код, що буде вирішувати необхідну задачу, у цьому випадку - виведення (по команді) на екран рядка тексту "Hello World!".

- Виберіть на формі першу із кнопок і перейдіть в інспекторі об'єктів на сторінку подій (Events).

- Двічі натисніть кнопку миші на події `OnClick`; середовище створить заготовку процедури обробки `Button1Click()` і встановить курсор редактора коду усередині її. Уведіть рядок коду, щоб функція виглядала так:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Label1->Caption = "Hello World!";
}
```

- Так само створіть процедуру обробки події для другої командної кнопки:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Form1->Close();
}
```

- При бажанні змініть розмір форми (як це робиться зі звичайним вікном) і збережіть файли проекту.
- Необхідно відкомпілювати й запустити програму. Для виконання натисніть кнопку із зеленою стрілкою (Run).
- Натисніть кнопку `Message`. Буде виведений рядок повідомлення. Кнопка `Exit` завершує роботу програми, закриваючи її головне вікно (його можна закрити й системною кнопкою в правому верхньому куті - це те ж саме).

Таким чином, процес візуального проектування користувальницького інтерфейсу полягає в тому, що з палітри вибираються необхідні компоненти, розміщаються на формі, підганяються по положенню, розміру; для них установлюються різні властивості за допомогою інспектора об'єктів. Після розміщення всіх необхідних компонентів потрібен етап програмування, тобто написання коду для різних подій. Варто звернути увагу, що у візуально спроектованому додатку будь-який написаний програмістом код так чи інакше викликається з деякої процедури обробки події. У нашому проекті ми ввели програмний код у функції `OnClick()` для обох командних кнопок, це обробник події `Click`. Перша з них привласнює властивості `Caption` розташованої на формі мітки необхідний символічний рядок. Друга просто викликає функцію закриття форми `Close()`, це метод.

Команди роботи із проектом

Збереження проекту

Якщо проект жодного разу не зберігався, для його збереження варто вибрати команду головного меню **File|Save Project As** (Файл|Сохранить як). По цій команді виводиться вікно збереження модуля (`Unit1`), а потім - вікно збереження головного модуля (`Project1`). У ході діалогів первісні імена файлів, що дають середовищем Borland Delphi за замовчуванням (`Project1`, `Unit1`, `Unit2` і т.д.), рекомендовано перейменувати.

Якщо проект уже був збережений, то повторне збереження при змінах варто виконувати такими способами:

- за допомогою команди головного меню **File|Save All** (Файл|Сохранить весь);
- за допомогою кнопки **Save All** панелі інструментів;
- за допомогою команди **Save** локального меню адміністратора проектів.

У всіх випадках зберігаються файли проекту, у яких відбулися зміни (у тексті або формі). У перших двох випадках зберігаються також змінені файли, що не входять у проект.

Можна зберегти проект і під іншим ім'ям. Це може знадобитися, якщо створюється новий варіант програми. Для цього варто вибрати команду головного меню **File|Save Project As** (Файл|Сохранить як). По цій команді виводиться вікно вибору імені файлу, у якому треба задати нове ім'я. Якщо файл із таким ім'ям уже є, буде видане попередження про це. Перейменовується тільки програма-проект (якщо тільки в проекті немає файлів, жодного разу не зберігалися).

Збереження файлу

Щоб зберегти відкритий модуль у файлі, можна скористатися командою головного меню **File|Save** або кнопкою **Save** панелі інструментів. Попередньо варто зробити активним текст модуля, що зберігається, або його форму, якщо вона є.

Для збереження файлу під іншим ім'ям цього досить скористатися командою головного меню **File|Save As**. При виконанні цієї команди виводиться вікно вибору імені файлу для завдання нового імені. Після вибору нового імені зміниться на нове також ім'я модуля, а в програмі-проекті також на нову зміниться зв'язок зі старим файлом.

Відкриття існуючого проекту

Для того щоб відкрити існуючий проект, можна скористатися наступними можливостями:

- використати команду головного меню **File|OpenProject** ;

Якщо проект, якому варто відкрити, був недавно закритий, можна скористатися командою головного меню **File|Reopen** (Файл|Снову відкрити), що дозволяє відкрити один з декількох останніх закритих проектів.

Додавання до проекту нового файлу

До проекту можна додавати файли, що містять різні модулі, включаючи модулі форми, дані й т.д.

Найбільш повні можливості тут при використанні команди головного меню **File|New**, що відкриває вікно вибору елементів проекту. Аналогічний результат можна одержати, нажавши кнопку **New** адміністратора проектів.

Модуль, що містить форму, можна підключити, використовуючи команду головного меню **File|New Form** або кнопку панелі інструментів **New Form**. При додаванні нової форми створюється заготівля її тексту, зображення порожньої форми, а сама вона підключається до проекту в програмі-проекті.

Додавання до проекту існуючого файлу

Для додавання до проекту існуючого файлу з тим або іншим модулем можна скористатися наступними можливостями:

- використати команду головного меню **Project|Add to Project** (Проект|Добавить до проекту);
- нажати кнопку **Add file to project** панелі інструментів;
- використати команду **Add** локального меню адміністратора проектів.

У всіх цих випадках відкривається вікно вибору імені файлу для завдання імені файлу, що підключають, (модулі перебувають у файлах з розширенням.cpp).

7. Видалення файлу із проекту

Для видалення файлу із проекту можна використати наступні можливості:

- використати команду головного меню **Project|Remove from Project** (Проект|Удалити із проекту);
- натиснути кнопку **Remove file from project** панелі інструментів;
- використати команду **Remove File** локального меню адміністратора проектів.

Відкриття файлу

Можна відкрити файл, не включаючи його в проект (наприклад, для того, щоб скопіювати його яку-небудь частину у файли проекту, або для порівняння його з файлами проекту), або відкрити файл, уже включений у проект, але за якимись причинами закритий. Файл може містити програму-проект, модуль форми, модуль або текст. Відкрити файл можна:

- за допомогою команди головного меню **File|Open** (Файл|Відкрити);
- натисканням кнопки **Open** панелі інструментів.

У цих випадках відкривається вікно вибору імені файлу, за допомогою якого вибирається ім'я файлу, що відкривається.

Якщо файл, що потрібно відкрити, був недавно закритий, його можна відкрити за допомогою команди головного меню **File|Reopen**.

Якщо потрібно відкрити файл, включений у проект, можна скористатися кнопками панелі інструментів перегляду тексту файлу - **View Unit** (Перегляд Елемента) або перегляду форми - **View Form** (Перегляд Форми), якщо вона в модуля є. Попередньо варто виділити мишею або за допомогою клавіатури відповідний рядок у списку модулів проекту. При цьому на екрані відкривається як текст модуля, так і його форма. Цю ж операцію можна виконати або активізувавши мишею ім'я модуля або ім'я форми модуля в списку модулів проекту, або активізувавши клавіатурою відповідний рядок у цьому списку. У кожному разі на екрані з'являються й текст модуля, і форма.

Відкриття програми-проекту (головного модуля)

При відкритті будь-якого файлу проекту проект-програма-проект на екран не виводиться. Для того щоб переглянути її текст, варто скористатися командою головного меню **Project|View Source** (Проект|Просмотр источника). Текст програми-проекту з'являється на новій сторінці текстового редактора.

Закриття проекту

Закрити проект можна такими способами:

- використати команду головного меню **File|Close All** (Файл|Закрити все);
- відкрити іншої (може бути, новий) проект;
- закрити програму-проект.

У всіх випадках закриваються всі відкриті файли (у тому числі й не вхідні в проект).

Закриття файлу

Для того щоб закрити файл, можна скористатися командою головного меню **File|Close**. Попередньо варто активізувати той модуль (або проект проект-програма-проект), якому потрібно закрити, зробивши активним або його текст, або форму, якщо вона є в модуля. При виконанні команди відповідний модуль закривається, при цьому віддаляються з екрана його текст і форма, якщо вона є.

Для зміненого модуля попередньо видається повідомлення, чи варто зберегти його зміни. Сам модуль із проекту не віддаляється.

Структура простого проекту

Проект являє собою набір програмних одиниць - модулів.

Один з модулів, називаний головним, містить інструкції, з яких починається виконання програми. Щоб побачити головний модуль, потрібно в меню Project вибрати команду View Source. Як приклад у лістингу 1 наведений текст головного модуля програми "Hello, World!".

Лістинг 1. Проект

```
#include <vcl.h>
#pragma hdrstop
#include <tchar.h>
USEFORM("Unit1.cpp", Form1);
//-----
int WINAPI _tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int)
{
    try
    {
        Application->Initialize();
        Application->MainFormOnTaskBar = true;
        Application->CreateForm(__classid(TForm1), &Form1);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    catch (...)
    {
        try
        {
            throw Exception("");
        }
        catch (Exception &exception)
        {
            Application->ShowException(&exception);
        }
    }
    return 0;
}
```

Починається головний модуль директивами препроцесору щодо підключення ресурсів. Рядок указує, що в проект потрібно включити файл модуля форми Unit1, що містить функції обробки подій для форми Form1. Далі йде код, який ініціалізує внутрішні структури програми, створює форму Form1 і запускає програму, що приводить до появи на екрані стартової форми. Тому що в проекті "Hello, World!" тільки одна форма, то на екрані саме вона й з'являється.

Листинг 2. Файл модуля Unit1.h

```
class TForm1 : public TForm
{
__published:    // IDE-managed Components
    TButton *Button1;
    TLabel *Label1;
    void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:    // User declarations
    __fastcall TForm1(TComponent* Owner);
};
//-----
extern PACKAGE TForm1 *Form1;
```

Слід зазначити, що значну кількість роботи з генерації програмного коду виконано середовищем. Повністю сформувано головний модуль, файл модуля форми. Файл, що містить опис форми Unit1.dfm, можна переглянути в редакторі коду, клацнувши на формі правою клавішею миші й вибравши в контекстному меню команду View as text. Перейти назад на форму можна за допомогою команди контекстного меню View as Form.

```
//опис форми Unit1.dfm
object Form1: TForm1
    Left = 0
    Top = 0
    Caption = 'Form1'
    ClientHeight = 202
    ClientWidth = 447
    Color = clBtnFace
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = []
    OldCreateOrder = False
    PixelsPerInch = 96
    TextHeight = 13
    object Label1: TLabel
        Left = 16
        Top = 16
        Width = 31
        Height = 13
        Caption = 'Label1'
    end
    object Button1: TButton
        Left = 72
        Top = 8
        Width = 75
        Height = 25
        Caption = 'Button1'
        TabOrder = 0
    end
end
```

```
OnClick = Button1Click  
end  
end
```

Додати на форму компонент Edit, задати в інспекторі об'єктів йому властивість Name=Edit1. Аналогічно для кнопки Button1 задати властивість Caption “Обробити”.

В інспекторі об'єктів на сторінці Events задати обробник події OnClick подвійним натисканням клавіші миші та ввести текст

```
Label1->Caption=Edit1->Text;
```

Зберегти проект. Виконати проект.

Оформити звіт. У звіті скорочено на сторінку своїми словами описати інструментарій середовища розробки. Далі описати файли проекту. Описати послідовність команд, використаних для розробки програми. Прокоментувати код програми на предмет «компонент», «властивість», «метод», «подія», навести скріншот екрану під час виконання програми.

ЛАБОРАТОРНА РОБОТА № 3-4

Розробка та відлагодження програми з використанням одно- та багаторядкового текстового редактору, групування та розмеження доступу

Мета: вивчення прийомів розробки та відлагодження програми з використанням одно- та багаторядкового текстового редактору, групування та розмеження доступу

Частина 1

Постановка задачі. Проведення розрахунків виразів у відповідності з формулами. На формі створити два GroupBox, в одному реалізувати завдання перетворити поточне значення температури, задане в градусах Цельсія, в значення за Фаренгейтом та свій варіант перетворення; в другому - обчислити значення струму в електричному ланцюгу, якщо задані напруга та опір.

Порядок виконання

Проектування інтерфейсу.

Для GroupBox1 виконати наступне.

Розмістити два надписи «Температура» та «Результат», для введення однорядковий редактори (TEdit), для якого встановити властивість Name edtTemperature і очистити властивість Text. Для виведення результату використати компонент Panel. Встановити його властивість Name в pnlResult, Caption (заголовок) - рівним "Нуль", а BevelOuter (зовнішній скіс) - bvLowered (нижній). Для форми Caption (заголовок) - "Перетворення температури" і BorderStyle - bsSingle (поодинокий, пояснити, що він дає).

Розробка функцій інтерфейсу.

Програма повинна працювати за допомогою події OnChange для компонента Edit. Ця подія настає при зміні тексту, що відображається в компоненті Edit. Згенерувати обробник події OnChange - функцію з ім'ям edtTemperatureChange. Зразок коду:

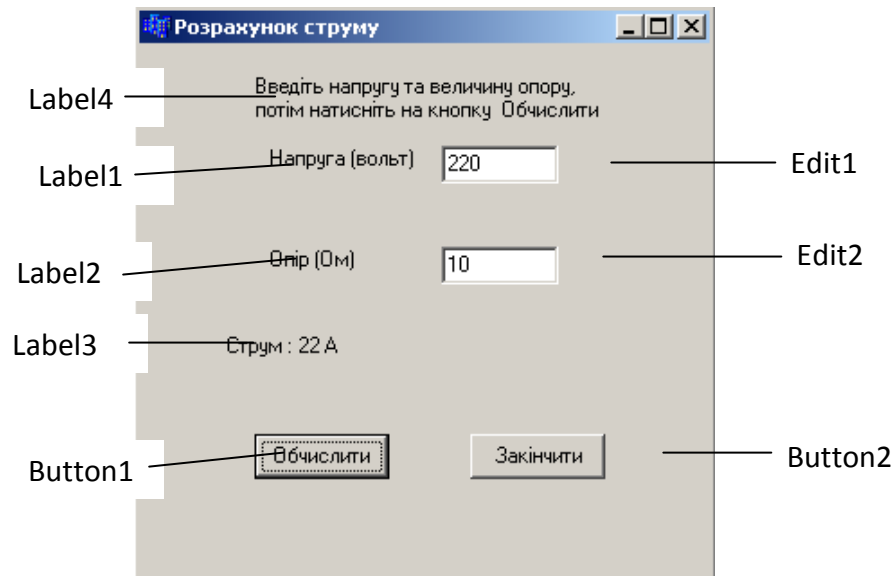
```
int iCelsius;
iCelsius = StrToIntDef(edtTemperature->Text, 0);
pnlResult->Caption = FloatToStr(iCelsius * 1.8 + 32);
```

Функція StrToIntDef дозволяє вказати значення, яким треба скористатися у випадку, якщо рядок не надасть допустимого цілого значення (останній параметр функції). Оскільки результат перетворення виражається в числах з плаваючою точкою, необхідно використовувати функцію FloatToStr для його перетворення в текстове значення, потрібне властивістю Caption компонента Panel.

Додати інші компоненти Edit, Button і виконати розрахунок згідно варіанту. Замість події OnChange для edit використати подію натискання на кнопку button

№ в	Призначення програми	Співвідношення для перерахунку
1	Перерахунок швидкості вітру з м/с в км/год	км/год = м/с*3.6
2	Перерахунок швидкості вітру з км/год в м/с	км/год = м/с*3.6
3	Перерахунок маси з фунтів в кілограми	1 фунт = 409.5 г
4	Перерахунок маси з кілограмів у фунти	1 фунт = 409.5 г
5	Перерахунок кутових градусів в радіани	$360^{\circ} = 2\pi$ радіан
6	Перерахунок радіан в кутові градуси	$360^{\circ} = 2\pi$ радіан
7	Перерахунок довжини з метрів у фути	1 фут = 0.305 м
8	Перерахунок довжини з футів в метри	1 фут = 0.305 м
9	Перерахунок довжини з дюймів в сантиметри	1 дюйм = 2.54 см
10	Перерахунок довжини з сантиметрів в дюйми	1 дюйм = 2.54 см
11	Перерахунок кількості теплоти з калорій в джоулі	1 калорія = 4.19дж
12	Перерахунок кількості теплоти з калорій в ерги	1 калорія = $4.19 \cdot 10^7$ ергів
13	Перерахунок кількості теплоти з джоулів в калорії	1 калорія = 4.19дж
14	Перерахунок довжини з сантиметрів в дюйми	1 дюйм = 2.54 см
15	Перерахунок кількості теплоти з ергів в калорії	1 калорія = $4.19 \cdot 10^7$ ергів
16	Перерахунок швидкості вітру з м/с в км/год	км/год = м/с*3.6
17	Перерахунок швидкості вітру з км/год в м/с	км/год = м/с*3.6
18	Перерахунок маси з фунтів в кілограми	1 фунт = 409.5 г
19	Перерахунок маси з кілограмів у фунти	1 фунт = 409.5 г
20	Перерахунок кутових градусів в радіани	$360^{\circ} = 2\pi$ радіан
21	Перерахунок радіан в кутові градуси	$360^{\circ} = 2\pi$ радіан
22	Перерахунок довжини з метрів у фути	1 фут = 0.305 м
23	Перерахунок довжини з футів в метри	1 фут = 0.305 м
24	Перерахунок довжини з дюймів в сантиметри	1 дюйм = 2.54 см
25	Перерахунок довжини з сантиметрів в дюйми	1 дюйм = 2.54 см
26	Перерахунок кількості теплоти з калорій в джоулі	1 калорія = 4.19дж
27	Перерахунок кількості теплоти з калорій в ерги	1 калорія = $4.19 \cdot 10^7$ ергів

Для GroupBox2 виконати наступне, зразок нижче



Розмістити на формі компоненти Label, Edit, Button у відповідності з тим, як показано на рисунку. Змінити у відповідності з рисунком властивість Caption для міток, кнопок та форми та очистити властивість Text для компонентів Edit.

Перейти до редактору коду і в обробник OnClick для кнопки "Обчислити" необхідно додати програмний код, в результаті чого функція буде мати вигляд:

```
float u, i, r; //напруга, струм, опір
if ( (Edit1->Text.Length()==0) || (Edit2->Text.Length()==0) ) {
    MessageDlg("Треба ввести струм та опір ",
mtInformation, TMsgDlgButtons () << mbOK, 0);
    if (Edit1->Text.Length() == 0) {
        Edit1->SetFocus(); // курсор в поле "Напруга"
    } else {
        Edit2->SetFocus(); // курсор в поле "Опір"

        return;
    }
}
u=StrToFloat(Edit1->Text);
r=StrToFloat(Edit2->Text);
i=u/r;
Label3->Caption=FloatToStrF(i,ffGeneral,7,2) + ' А ';
```

Друга кнопка для завершення програми, додати програмний код
Form2->Close();

Для поля «Напряжение» (Edit1) організувати обробник Edit1KeyPress і додати в нього наступний програмний код:

```
// нажатие клавиши в поле Напряжение
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
    // Коды заборонених клавіш замінюються на 0, в результаті
    // символи цих клавіш в полі редагування не 'являться
    // Key — код клавіші, що натиснута
    // перевіримо, чи є символ допустимим
    if ( ( Key >= '0') && ( Key <= '9')) // цифра
```

```

return;
// Глобальна змінна DecimalSeparator
// содержит символ, используемый в качестве разделителя
// при записи дробных чисел
if ( Key == DecimalSeparator) {
if ( Edit1->Text.AnsiPos(DecimalSeparator) != 0) {
Key = 0; // разделитель уже введен
}
return;
}
if (Key == 8) // <Backspace>
return;
if ( Key == 13) // <Enter>
{
Edit2->SetFocus() ;
return;
}
// остальные клавиши запрещены
Key = 0; // не отображать символ
}

```

Частина II

Постановка задачі. В проєкті передбачити два контейнера для реалізації таких двох задач:

1 Проведення розрахунків у відповідності з формулами з використанням багаторядкових текстових редакторів (наприклад, ТМемо) для введення і виведення. Ввести масив А цілих чисел, одержати новий масив В, замінивши у вихідному масиві елементи, що стоять на парних місцях, їхніми квадратами (по натисканню на кнопку). Передбачити другу кнопку для обчислення завдання за варіантом з таблиці 2.

2 Обчислення середнього заробітку працівника, якщо відомо його щомісячну оплату за інтервал часу, не більший 12 місяців. Рекомендовані компоненти наведені в таблиці 3. Передбачити обмеження доступу (наприклад, з використанням властивості Enabled).

Порядок виконання

Розглянемо варіант організації введення масиву у рядки компонента Мемо. Основні властивості компонента Мемо наведені в таблиці 1.

Таблиця 1 - Основні властивості компонента Мемо

Властивість	Визначає
Name	Ім'я компонента - для доступу до властивостей компонента
Text	Текст, що перебуває в полі Мемо. Розглядається як єдине ціле
Lines.Strings[i]	Текст, що перебуває в i-му рядку. Нумерація рядків з нуля
Lines.Count	Кількість рядків тексту в полі Мемо
Left	Відстань від лівої границі поля Мемо до лівої границі форми
Top	Відстань від верхньої границі до верхньої границі форми
Height	Висота поля Мемо
width	Ширина поля Мемо

Font	Шрифт, використовуваний для відображення тексту, що вводиться
ParentFont	Ознака спадкування властивостей шрифту батьківської форми

При використанні компонента Мемо для введення масиву значення кожного елемента масиву потрібно вводити в окремому рядку.

Lines - властивість компонента Мемо, що представляє собою масив, кожний елемент якого містить один рядок тексту Мемо. Одержати доступ до тексту, що знаходиться в рядку, можна вказавши у квадратних дужках номер потрібного рядка (нумеруються з 0).

У програмі для виводу повідомлень використовується функція ShowMessage, що виводить на екран вікно з текстом і командною кнопкою ОК. Інструкція виклику функції ShowMessage виглядає так:

```
ShowMessage("Текст повідомлення");
```

У заголовку вікна повідомлення, виведеного функцією ShowMessage, вказується назва додатка. Назва додатка задається на вкладці Application вікна Project Options. Якщо назву додатку не задано, то в заголовку буде ім'я файлу, що виконується.

В лістингу 1 наведено фрагмент коду програми, що демонструє використання компонента Мемо для введення масиву цілих чисел. Для створення інтерфейсу програми розмістити на формі два компоненти Мемо1 і Мемо2 а також кнопку (Button1). Над компонентом Мемо1 розмістити надпис «Вхідний масив», над компонентом Мемо2 розмістити надпис «Вихідний масив». При натисканні на кнопку виконується введення значень елементів до масиву А з Мемо1, потім відповідно до поставленого завдання розраховуються елементи масиву В і виводяться в Мемо2.

Лістинг 1 - Введення масиву цілих чисел

```
const int size = 5; // розмір масиву
int A[size], B[size]; // масиви
int i, n; //індекс елемента масиву та кількість рядків, введених у
поле Мемо
String st;
n = Memo1->Lines->Count;
if (n == 0) {
    ShowMessage ("Вхідні дані не введені!");
    return; // вихід із процедури обробки події
}
if (n > size) { // у полі Memo1 є текст
    ShowMessage ("Кількість рядків перевищує розмір масиву");
    n = size; // будемо вводити тільки перші SIZE рядків
}
for (i=0; i<n; i++)
    A[i] = StrToInt(Memo1->Lines->Strings[i]); // рядки Мемо
    пронумеровані з нуля
for (i=1; i<=n; i++) { //створення нового масиву
    if ((i % 2) == 0) {
        B[i-1] = pow(A[i-1], 2);
    } else {
        B[i-1] = A[i-1];
    }
}
```

```

}
for (i=0; i<n; i++) // виведення масива В в Мемо2
    Memo2->Lines->Add(IntToStr(B[i]));

```

Додати на форму іншу кнопку і з тими ж компонентами виконати розрахунок згідно варіанту. Очистити Мемо можна як Memo1->Clear();

Виконати за натисканням другої кнопки розрахунок за варіантом. Якщо якийсь елемент масиву А не може бути перетворений у елемент масиву В, то цей елемент не розраховується. Для тих варіантів, де потрібний діапазон, вводити з форми.

Таблиця 1 - Варіанти завдань для обчислення масивів

N варіанту	У масив А вводяться	У масив В виводяться
1	Цілі числа	Елементи масиву А, які лежать у заданому діапазоні [n1, n2]
2	Дійсні числа	Корені квадратні з елементів масиву А
3	Цілі числа	Додатні елементи масиву А
4	Дійсні числа	Від'ємні елементи масиву А
5	Цілі числа	Елементи масиву А, які не лежать у заданому діапазоні [n1, n2]
6	Дійсні числа	1/A[i]
7	Цілі числа	Тільки ненульові елементи масиву А
8	Дійсні числа	Додатні та нульові елементи масиву А
9	Цілі числа	Елементи масиву А, які лежать у заданому діапазоні [n1, n2]
10	Дійсні числа	Додатні та від'ємні елементи масиву А
11	Цілі числа	Від'ємні та нульові елементи масиву А
12	Дійсні числа	Корені квадратні з елементів масиву А, помножених на -1
13	Цілі числа	Елементи масиву А, які лежать у заданому діапазоні [n1, n2]
14	Дійсні числа	Корені квадратні з елементів масиву А
15	Цілі числа	Додатні елементи масиву А
16	Дійсні числа	Від'ємні елементи масиву А
17	Цілі числа	Елементи масиву А, які не лежать у заданому діапазоні [n1, n2]
18	Дійсні числа	1/A[i]
19	Цілі числа	Тільки ненульові елементи масиву А
20	Дійсні числа	Додатні та нульові елементи масиву А
21	Цілі числа	Елементи масиву А, які лежать у заданому діапазоні [n1, n2]
22	Дійсні числа	Додатні та від'ємні елементи масиву А
23	Цілі числа	Від'ємні та нульові елементи масиву А
24	Дійсні числа	Корені квадратні з елементів масиву А, помножених на -1
25	Цілі числа	Елементи масиву А, які лежать у заданому діапазоні [n1, n2]
26	Дійсні числа	Корені квадратні з елементів масиву А
27	Цілі числа	Додатні елементи масиву А

Для реалізації завдання обчислення середнього заробітку працівника для розробки інтерфейсу будемо використовувати компоненти, перелік яких наведений у таблиці.

Таблиця 2 - Рекомендовані компоненти

Компонент (и) або властивості	Призначені для
GroupBox1	групування компонентів, призначених для введення тривалості періоду розрахунку
GroupBox2	групування компонентів, призначених для введення щомісячної заробітної плати за періоду розрахунку
GroupBox3	виділення області виводу результатів
Edit1, Edit2	введення тривалості інтервалу й величини заробітної плати
Label1, Label2	виводу пояснювального тексту
Label3	виводу значення середнього заробітку
Button1, Button2, Button3	активізації процедур оброблювачів подій

Зразок розташування компонентів показаний нижче.

Значення деяких властивостей компонентів наведені в таблиці.

Властивість	Значення
GroupBox1.Color	dAqua
GroupBox2.Color	dYellow
GroupBox3.Color	dMoneyGreen
Label1 .Caption	'Введіть тривалість інтервалу не більше 12 місяців'
Label1 .WordWrap	True
Label2 .Caption	'Введіть зарплату'
Label3 .WordWrap	True
Label3. Caption	'Значення середньої зарплати'

Після виходу на екран форми програма повинна заборонити введення зарплати, поки не введена тривалість розрахункового інтервалу. Після цього

зробити доступними тільки ті компоненти, які призначені для введення місячної зарплати.

Для цього створимо для форми оброблювач події OnCreate (створення форми) а також оброблювачі події OnClick для кнопок. Внесемо в них код, подібний до наведеного нижче.

```
...FormCreate
...
    GroupBox1->Enabled=true;
    GroupBox2->Enabled=false;
    GroupBox3->Enabled=false;
//-----
...Button1Click
...
    GroupBox1->Enabled=false;
    Dlit_Int=StrToInt(Edit1->Text);
    Nom_Tek_Mes=0;
    Label2->Caption:="Введіть для місяця "+IntToStr(Nom_Tek_Mes+1);
    GroupBox2->Enabled=true;
    Edit2->SetFocus();
//-----
-
...Button2Click
...
    Mes_Zarpl[Nom_Tek_Mes]=StrToFloat(Edit2->Text);
    Nom_Tek_Mes=Nom_Tek_Mes+1;
    if (Nom_Tek_Mes<Dlit_Int) {
        Label2->Caption:="Введіть          для          місяця
        "+IntToStr(Nom_Tek_Mes+1);
        Edit2->Text="";
        Edit2->SetFocus();
    } else {
        GroupBox2->Enabled=false;
        GroupBox3->Enabled=true;
        Sred_Zarpl=0;
        for (i=0; i<Dlit_Int; i++)
            Sred_Zarpl=Sred_Zarpl+Mes_Zarpl[i];
        Sred_Zarpl=Sred_Zarpl/Dlit_Int;
        Label3->Caption=FloatToStr(Sred_Zarpl);
    }
//-----
...Button3Click
...
    Form1.Close();
//-----
```

В секції private для форми можна додати змінні:

```
int Nom_Tek_Mes, Dlit_Int;
float Mes_Zarpl[12];
float Sred_Zarpl;
```

Виконайте програму. Поясніть її роботу. Оформіть звіт.

ЛАБОРАТОРНА РОБОТА № 5-6

Розробка програми з використанням списків ListBox. Використання рядка стану

Мета: набуття навичок розробки програми з використанням списків ListBox. Використання рядка стану

Частина I

Постановка завдання.

Розробити програму обчислення вартості міжміської телефонної розмови, якщо відомі вартість однієї хвилини розмови для країни-абонента, величина знижки на вартість розмови по телефону у вихідні дні. Країну, день тижня і тривалість розмови задає користувач. Вартість однієї хвилини розмови задана в таблиці.

	Країна, з якою ведеться розмова	Вартість однієї хвилини розмови
1	Україна	0.6 грн.
2	Росія	1.2 грн.
3	Англія	2.5 грн.
4	Германія	2.8 грн.

Розробка інтерфейса

Для розробки інтерфейсу використовуватимемо компоненти ListBox, Label, Edit, Button, UpDown, GroupBox, призначення яких приведене в таблиці.

У таблицях перераховані компоненти форми додатка а також наведені значення властивостей компонентів.

“Компоненти форми додатка”

Компонент(и) або властивості	Призначен(і) для
GroupBox1	групування компонентів, що забезпечують введення початкових даних
GroupBox2	виділення області виведення результатів
Edit1	введення тривалості розмови в хвилинах
UpDown	введення номера дня тижня
UpDown.Position	зберігання значення номера дня тижня
Label1, Label2, Label3	висновку пояснювального тексту про призначення елементів інтерфейсу
Label4	виведення значення вартості розмови
ListBox	вибору країни, з якою ведеться телефонна розмова
Button1, Button2	активізації процедур обчислення вартості розмови і завершення програми

Зразок розташування компонентів на формі показаний нижче.

Властивість	Значення
GroupBox1.Color	dAqua
GroupBox2.Color	dMoneyGreen
UpDown.Associate	Edit2
UpDown.Min	1
UpDown.Max	7
UpDown.Increment	1
ListBox1.Sorted	True

Для програми, що розробляється, найбільший інтерес представляють дві властивості: Items і ItemIndex.

Властивість Items містить елементи списку. Список, що виводиться в полі списку, може бути сформований на етапі проектування додатка або динамічно, під час роботи програми. Для формування списку під час створення форми додатка потрібно у формі виділити поле списку, потім у вікні Object Inspector вибрати властивість Items і клацнути по кнопці редактора списку рядків, на якій зображено три точки.

У діалогове вікно String, що з'явилося, list editor ввести з клавіатури в окремих рядках найменування країн, перерахованих в таблиці (вартість хвилини розмови не вводити).

В таблиці наведені властивості компонента ListBox

Name	Ім'я компонента, У програмі використовується для доступу до властивостей компонента
Items	Елементи списку
Itemindex	Номер обраного елемента списку. Номер першого елемента списку дорівнює нулю
Left	Відстань від лівої границі списку до лівої границі форми
Top	Відстань від верхньої границі списку до верхньої границі

	форми
Height	Висоту поля списку
Width	Ширину поля списку
Font	Шрифт, використовуваний для відображення елементів списку
ParentFont	Ознака спадкування властивостей шрифту батьківської форми
Sorted	Ознака сортування списку

Найбільший інтерес представляють властивості Items і ItemIndex. Властивість Items містить елементи списку.

Властивість ItemIndex задає номер обраного елемента списку. Якщо жоден з елементів не обраний, то значення властивості дорівнює мінус одиниці.

Список може бути сформований під час створення форми або під час роботи програми.

Для формування списку під час створення форми треба у вікні Object Inspector вибрати властивість Items і клацнути на кнопці запуску редактора списку рядків (кнопка маркірована трьома крапками).

У діалоговому вікні, що відкрилося, String List Editor потрібно набрати список, кожний елемент списку в окремому рядку. Після уведення чергового елемента списку, для переходу до нового рядка, необхідно натиснути клавішу <Enter>. Після уведення останнього елемента списку клавішу <Enter> натискати не треба. Завершивши уведення списку, варто клацнути на кнопці ОК.

Програма повинна виконувати обчислення при щиглику на кнопці "Розрахувати вартість розмови". Зразок програмного коду методу мовою Pascal приведено нижче. Оформити програмний код мовою C++. Коментарі та надписи виконати українською.

```
//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    const int DISCOUNT(20); //знижка 20 відсотків
    String Strana;
    double Time; //кількість хвилин розмови
    double Pay; //ціна однієї хвилини розмови
    int Day; //день тижня
    double Summa; //вартість розмови
    //отримати вказані дані
    if (ListBox1->ItemIndex < 0) { ShowMessage("Вкажіть країну"); }
    else { //враховано, що список відсортовано
        switch (ListBox1->ItemIndex) {
            case 0: { Pay = 2.5; break; }
            case 1: { Pay = 2.8; break; }
            case 2: { Pay = 1.2; break; }
            case 3: { Pay = 0.6; break; }
            default: { ShowMessage("Країни немає в списку"); }
        }
        Strana = ListBox1->Items->Strings[ListBox1->ItemIndex];
    }
    Time = StrToFloat(Edit1->Text);
```

```

Day = UpDown1->Position;
ShowMessage("Day =" + IntToStr(Day));
//обчислення вартості розмови
Summa = Pay*Time;
//В суботу або неділю вартість зменшено відповідно до знижки
if ((Day == 6) || (Day == 7)) { Summa = Summa * (100-
DISCOUNT)/100; }
//виведення результату обчислення
Label4->Caption = "Країна - "+Strana+". До сплати -
"+FloatToStr(Summa) + " грн.";
}
//-----

```

Частина II

Постановка завдання.

Доповнити програму частини 1 обчисленням підсумкових даних загальної вартості міжміських телефонних розмов з кожною країною. За вибором країни в списку надавати в другій панелі рядку стану (StatusBar) інформацію щодо назви країни і загальної вартості всіх розмов. В першій панелі стану вивести власне прізвище.

Розробка інтерфейса.

Для розробки інтерфейсу використати компонент StatusBar. Він складається з набору панелей типу *TStatusPanels*, що відтворюють рядок стану. Розташовується, як правило, внизу форми. Основні властивості наведені в таблиці нижче.

Таблиця - Основні властивості компоненту StatusBar

Властивість	Призначення
SimplePanel	при SimplePanel = true весь рядок стану є єдиною панеллю, інакше є набором панелей
SimpleText	при SimplePanel = true задає текст на єдиній панелі
Panels	при SimplePanel = false задає набір панелей. Виклик з інспектора об'єктів, контекстного меню або подвійним кліком. У вікні PanelsEditor можна додавати панелі або задавати властивості існуючим
SizeGrip	для набору панелей задає можливість користувачу змінювати їх розмір під час виконання
Text	текст конкретної панелі

Width	ширина конкретної панелі
-------	--------------------------

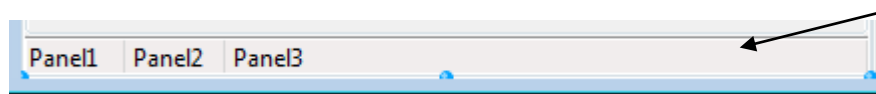
Програмно звернутися до конкретної панелі, наприклад, для виведення тексту, можна так:

```
StatusBar1->Panels->Items[0]->Text = "текст"; // для першої
```

Кількість панелей рядка стану визначається властивістю Count, наприклад:

```
for (int i = 0; i < StatusBar1->Panels->Count; i++) {
    StatusBar1->Panels->Items[i]->Text = "";
}
```

Зразок розташування на формі показаний нижче.



StatusBar1

Оформити звіт.

ЛАБОРАТОРНА РОБОТА № 7-8

Створення програми з використанням списків, що випадають. Використання файлу конфігурації

Мета: набуття навичок Створення програми з використанням списків, що випадають, використання файлу конфігурації

Частина I

Постановка завдання.

Магазин має певні товари, загальний список яких завантажується із створеного раніше текстового файлу або формується в процесі роботи програми. Загальний список може поповнюватись новими товарами з клавіатури. Новий список поповнюється за рахунок товарів, перелічених у загальному списку. Розробити програму, що дозволяє:

- 1 Завантажувати загальний перелік товарів з файлу або вводити.
- 2 Додавати або видаляти елементи загального списку.
- 3 Переносити дані із загального списку в новий список.
- 4 Зберігати, якщо потрібно, дані кожного списку у файлах.

Розробка інтерфейсу.

Для розробки інтерфейсу будемо використати компоненти RadioGroup, ComboBox, ListBox, Button, Label, OpenFileDialog, SaveDialog. Призначення деяких компонентів наведено в таблиці.

Компонент(и) або властивість	Призначення для
RadioGroup1	вибору способу формування загального списку
ComboBox1	відображення загального списку
ListBox	відображення нового списку

Button1,...	активізації відповідних процедур
Label1,...	пояснювальних написів
OpenDialog1	проведення діалогу відкриття файлу
SaveDialog1	проведення діалогів збереження файлів

Компонент ComboBox поєднує функції компонентів ListBox - списку й Edit - поля редагування. Цей компонент дозволяє вибрати зі списку необхідний рядок або задати як вибір власний текст.

Відмінність ComboBox і ListBox:

1 В ComboBox можна редагувати елементи списку, а в ListBox - не можна.

2 ComboBox може бути розгорнутий або згорнутий, а ListBox - завжди розгорнутий.

3 ListBox може допустити множинний вибір, а в ComboBox можна вибрати тільки один елемент.

4 ComboBox і ListBox можна заповнювати під час проектування або під час виконання програми.

5 ComboBox, як і ListBox має властивості Items і ItemIndex. Крім того, для компонента ComboBox визначена властивість Text, що містить текст, уведений у рядок введення.

Розташувати на формі компоненти, як зазначено на рисунку.

Для створення групи радіокнопок розмістити компонент RadioGroup1 і відредагувати його властивість Items у такий спосіб: увійти в його редактор і ввести два рядки, кожна з яких відповідає напису біля радіокнопки (кількість кнопок у радіогрупі відповідає кількості рядків у редакторі властивості Items).

Розташувати компоненти GroupBox1 і розмістити на ньому компоненти ComboBox1, ListBox1, кнопки Button1, Button2, Button3 і компоненти Label для пояснювальних написів.

Розмістіть компоненти діалогів і відповідні кнопки за зразком, як показано на рисунку 7.1.

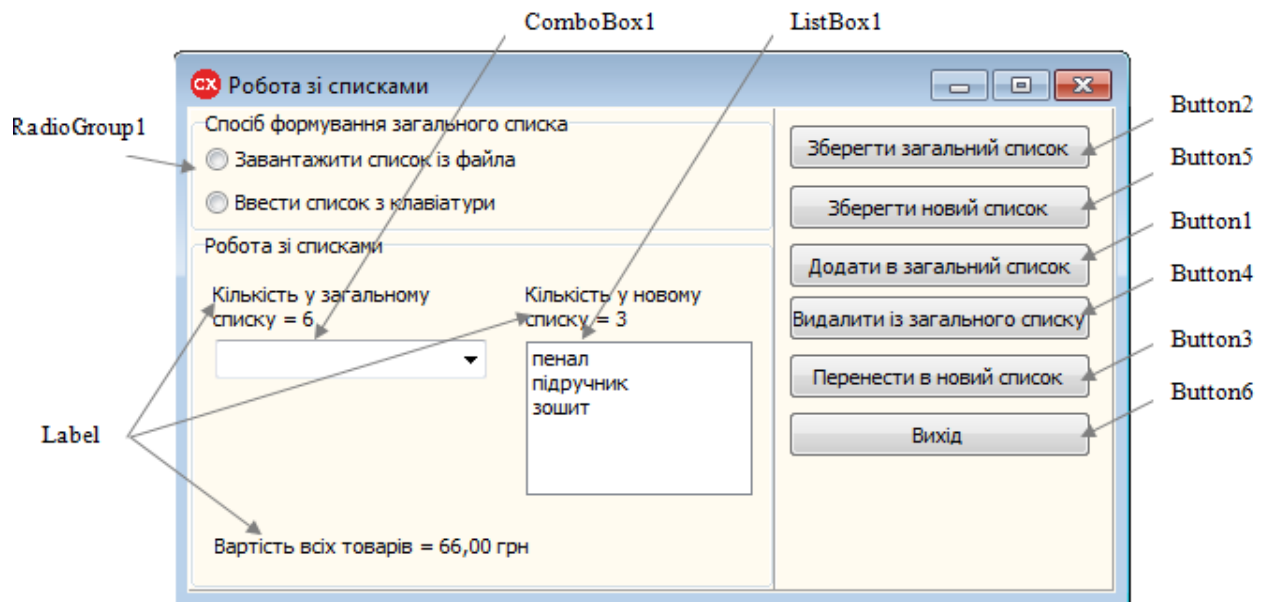


Рис.3.1 – Зразок розташування елементів

Задати компонентам властивості, значення яких наведені в таблиці.

Властивість	Значення
Form1->Caption	"Робота зі списком"
Button1->Caption	"Додати в загальний список"
Button4->Caption	"Видалити із загального списку"
Button3->Caption	"Перенести в новий список"
Button6->Caption	"Вихід"
Button2->Caption	"Зберегти загальний список"
Button5->Caption	"Зберегти новий список"
Button1->Hint	"Рядок, що додається, помістити в поле редагування"
Button1->ShowHint	true (підказка, що спливає)
Button4->Hint	"Рядок, що видаляється, помістити в поле редагування"
Button4->ShowHint	true

Мають бути створені наступні обробники подій (назви кнопок можуть бути інші):

```
... TForm1::Button6Click(...); // Кнопка "Вихід"
```

```
... TForm1::Button1Click(...); // Кнопка "Додати в загальний список"
```

```
... TForm1::Button2Click(...); // Кнопка "Зберегти загальний список"
```

```

... TForm1::FormCreate(...);
... TForm1::RadioGroup1Click(...);
... TForm1::Button3Click(...); // Кнопка "Перенести в новий список"
... TForm1::Button4Click(...); // Кнопка "Видалити із загального //списку"
... TForm1::Button5Click(...); // Кнопка "Зберегти новий список"

```

Додатково потрібно оголосити наступні змінні:

```

cost : integer;
ffname : string;

```

У обробники подій потрібно внести відповідний програмний код:

```

...
//-----
... TForm1::Button6Click(...)
{
    Close();
}
//-----
... TForm1::Button1Click(...)
{
    ComboBox1->Items->Add(ComboBox1->Text);
    Label2->Caption="Кількість в ComboBox1="+ IntToStr(ComboBox1->Items->Count);
}
//-----
... TForm1::Button2Click(...)
{
    if SaveDialog1->Execute()
        { ComboBox1->Items->SaveToFile(SaveDialog1->FileName); }
    else { ShowMessage("Загальний список не збережений"); }
}
//-----
... TForm1::FormCreate(...)
{
    GroupBox1->Enabled=false;
    Button2->Enabled=false;

```

```

        Button5->Enabled=false;
    }
//-----
... TForm1::RadioGroup1Click (...);
{
    if (RadioGroup1->ItemIndex==0) {
        if (OpenDialog1->Execute()) {
            ffname=OpenDialog1->FileName;
            ComboBox1->Items->LoadFromFile(ffname);
        }
    }
    GroupBox1->Enabled=true;
    Button2->Enabled=true;
    Button5->Enabled=true;
}
//-----
... TForm1::Button3Click (...)
{
    if (ComboBox1.Text != " ") {
        ListBox1->Items->Add(ComboBox1->Text);
        cost = cost + 22;
        Label1->Caption = "Вартість всіх товарів = " + IntToStr(cost)
+ ".00 грн.";
        Label3->Caption = "Кількість в ListBox1 =" + IntToStr(ListBox1-
>Items->Count);
        ComboBox1->ItemIndex = -1;
    }
}
//-----
... TForm1::Button4Click (...)
{
    ComboBox1->Items->Delete(ComboBox1->ItemIndex);
    ComboBox1->Text=" ";
    Label2->Caption="Кількість у ComboBox1 =" + IntToStr(ComboBox1-
>Items->Count);
}
//-----
... TForm1::Button5Click (...)

```

```

{
    if (SaveDialog1->Execute())
    {
        ListBox1->Items->SaveToFile(SaveDialog1->FileName); }
    {
        ShowMessage("Новий список не збережений"); }
    }
//-----

```

Пояснити етапи проєктування в звіті, не копіювати текст завдання до лабораторної роботи, описувати індивідуальний хід роботи.

Частина II

Постановка завдання.

Доповнити програму частини 1 використанням файлу конфігурації, ім'я файлу конфігурації — Prizvyshe.ini (підставити власне прізвище), розташування — за місцезнаходженням файлу проєкту або виконуваного файлу. В файлі конфігурації має бути секція "[Save]". При збереженні загального списку записувати в секцію "[Save]" як значення ідентифікатора Path той шлях, який користувач обрав компонентом SaveDialog в якості шляху збереження. При відкритті OpenFileDialog оброблювати подію OnShow і присвоювати властивості OpenFileDialog.InitialDir директорію з відповідного значення конфігураційного файлу.

Довідкова інформація по роботі з конфігураційним файлом наведена нижче.

Ini-файл — це текстовий файл з розширенням .ini, призначений для зберігання налаштувань програми. Текст в такому файлі згрупований по розділам, наприклад:

```

[Розділ1]
Ідентифікатор1=Значення1
Ідентифікатор2=Значення2
[Розділ2]
Ідентифікатор3=Значення3
...

```

Клас TIniFile оголошений в IniFiles, дописати #include <IniFiles.hpp>

Формат файлу INI загальноновживаний, багато конфігураційних файлів знаходяться в цьому форматі. Можна читати значення за допомогою різних методів, таких як, наприклад, `ReadString`. Зворотній метод для запису значення — `WriteString`. Нижче наведено приклад читання конфігураційної інформації з INI-файлу та запису значень.

Лістинг – Приклад читання конфігураційної інформації з INI-файлу

```
TIniFile *ini;  
ini = new TIniFile( "CONFIG.INI" );  
Form2->Caption = ini->ReadString ( "Form", "Caption", "Default  
Caption" );  
delete ini;
```

Метод `ReadString` приймає три параметри. Перший параметр визначає розділ INI-файлу. Другий параметр визначає ідентифікатор, який треба прочитати, а третій є значенням за замовчуванням на випадок, якщо розділ або ідентифікатор не існують у файлі INI.

Лістинг – Приклад запису в розділ "Form", ідентифікатор "Caption"

```
TIniFile *ini;  
ini = new TIniFile( "CONFIG.INI" );  
ini->WriteString ( "Form", "Caption", Form2->Caption );  
delete ini;
```

Оформити звіт, пояснити етапи проектування. Конфігураційний файл завантажити в архіві разом з файлами програми.

ЛАБОРАТОРНА РОБОТА № 9

Розробка програми з компонентом StringGrid

Мета: Відпрацювання прийомів розробки програми з використанням компоненту StringGrid

Частина I

Постановка завдання. Створити додаток, що використовує таблицю комірок StringGrid для введення чисел в першому стовпці та виведення розрахунку значень стандартних функцій в комірки таблиці.

Порядок проведення роботи

Створити новий проєкт, задати для форми властивість Caption “Обчислення стандартних функцій. Прізвище студента”, додати власне прізвище. Розташувати в нижній частині форми компонент Button1. При натисканні кнопки повинне вироблятися обчислення значення функцій для введеного аргументу, тому властивість Caption кнопки змінити на “Розрахувати”.

Зразок розташування елементів на формі наведений на рисунку 9.1.

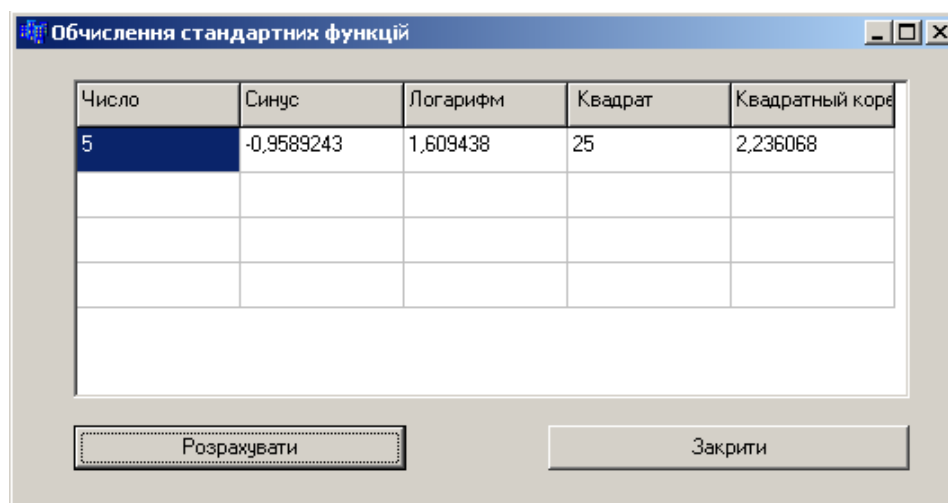


Рис.9.1 – Зразок розташування елементів на формі

Розмістити на формі таблицю комірок класу TStringGrid, вирівняти по верхньому краю (властивість Align=alTop). Задати властивості Options (опції) goEditing=True (можливість редагування в осередку таблиці). Задати також значення властивостей, наведені в таблиці 9.1.

Таблиця 9.1 – Додаткові властивості

Назва	Пояснення	Значення
ColCount	Кількість стовпців	5 (1 для параметра, що вводиться, і 4 для результатів)
DefaultColWidth	Ширина стовпців за замовчуванням	90 (ширина стовпця в пікселях)
FixedCols	Число фіксованих стовпців	0 (кількість фіксованих стовпців)
FixedRows	Кількість фіксованих рядків	1 (для заголовків)
RowCount	Кількість рядків	5

У верхньому (фіксованому) рядку таблиці розмістити написи стовпців, що вказують, яка стандартна функція розраховується для введеного значення, ці значення для кожного рядка передбачається вводити в першому стовпці. Як стандартні функції взяти функції, що обчислюють синус, логарифм, квадрат і квадратний корінь. Логарифма й квадратного кореня для негативних чисел не існує, тому обчислювати їх від аргумента по модулю.

Врахувати, що значення аргументів у таблиці (перший стовпець) можна коригувати, а значення функцій - ні. Передбачити можливість збільшувати при необхідності число рядків таблиці (коли всі наявні будуть заповнені).

Для форми в обробнику події OnCreate (при створенні) передбачити заповнення заголовків стовпців таблиці за зразком, наведеним в лістингу 9.1. Властивість Cells[i][j] (комірка) задає масив комірок таблиці, де перший індекс — номер стовпця, а другий індекс — номер рядка, індексація з 0.

Для введення аргументів функцій в першому стовпці можна скористатися подією таблиці OnSelectCell (виникає при виділенні комірки таблиці), де дозволити активізацію комірок у стовпці з індексом 0 і заборонити для інших стовпців, що наведено в лістингу 9.2. Параметр CanSelect (можна виділити) приймає значення true, якщо комірка таблиці може бути активізована, і false - якщо ні.

Програмний код обробника події кліку на кнопку “Розрахувати” наведений в лістингу 9.3. Для обчислення відповідних функцій як аргумент береться значення активної комірки першого стовпця (StringGrid1->Cells[0][StringGrid1->Row]), де StringGrid1 - ім'я таблиці, що виступає кваліфікатором, Cells - масив комірок таблиці, 0 - перший стовпець, StringGrid1->Row - активний рядок

таблиці), перетворюється з текстового начення в дійсне і обчислюється значення відповідної стандартної функції (sin, log10, power, sqrt), при необхідності, від абсолютного значення.

Якщо заповнений останній рядок таблиці, додається новий порожній рядок у таблицю для введення нового параметра.

Запустити програму й виділити мишею яку-небудь комірку першого стовпця, увести дійсне число і розрахувати. У результаті з'являться значення відповідних функцій для уведеного аргументу.

Уведення інформації можна повторювати багаторазово. Якщо буде уведена інформація в останній рядок таблиці, у неї з'явиться новий рядок. Якщо вся таблиця перестане вміщатися у формі, з'явиться вертикальна лінійка прокручування.

Відлагодити програму. Завершити роботу програми по натисканню кнопки “Закрити”.

Оформити звіт.

Лістинг 9.1 – Оброблювач події OnCreate форми

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    StringGrid1->Cells[0][0]="Число"; //Аргумент функцій
    StringGrid1->Cells[1][0]="Синус"; //Значення синуса
    StringGrid1->Cells[2][0]="Логарифм"; //Значення логарифма
    StringGrid1->Cells[3][0]="Квадрат"; //Значення квадрата
    StringGrid1->Cells[4][0]="Квадратний корінь"; //Значення квадратн. кореня
}
```

Лістинг 9.2 – Оброблювач події OnSelectCell таблиці

```
void __fastcall TForm1::StringGrid1SelectCell(TObject *Sender, int ACol,
int ARow, bool &CanSelect)
{
    if (ACol == 0) //Якщо обрано перший стовпець
    { CanSelect=true; } //Активізація дозволяється
    else //Інакше
    { CanSelect=false; } //Активізація забороняється
}
```

Лістинг 9.3 – Оброблювач натискання кнопки “Розрахувати”

```
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```

{
StringGrid1->Cells[1][StringGrid1->Row]=
FloatToStrF(sin(StrToFloat(StringGrid1->Cells[0][StringGrid1-
>Row])),ffGeneral,7,3);
StringGrid1->Cells[2][StringGrid1->Row]=
FloatToStrF(log10(abs(StrToFloat(StringGrid1->Cells[0][StringGrid1-
>Row]))),ffGeneral,7,3);
StringGrid1->Cells[3][StringGrid1->Row]=
FloatToStrF(pow(StrToFloat(StringGrid1->Cells[0][StringGrid1-
>Row]),2),ffGeneral,7,3);
StringGrid1->Cells[4][StringGrid1->Row]=
FloatToStrF(sqrt(abs(StrToFloat(StringGrid1->Cells[0][StringGrid1-
>Row]))),ffGeneral,7,3);
if (StringGrid1->Row + 1 == StringGrid1->RowCount) {
StringGrid1->RowCount=StringGrid1->RowCount + 1; }
}

```

ЛАБОРАТОРНА РОБОТА № 10

Створення програми з багатовіконним інтерфейсом

Мета: вивчення прийомів створення програми з багатовіконним інтерфейсом

Постановка задачі. Розробити програму, що вводить інформацію про рейси літаків: дати і часи відправлення, номери рейсів, пункти призначення. Введені дані повинні розміщуватися в масиві записів. Після введення початкової інформації про рейси користувач може отримувати довідки. Програма повинна видавати довідки про номери рейсів, які прямують в заданий пункт призначення і про пункти призначення рейсів, що відправляються в заданому інтервалі часу. Заголовки форм мають містити прізвище студента.

Розробка інтерфейсу

Програма повинна мати три вікна: головне вікно, з якого викликається решта вікон, вікно введення інформації і вікно отримання довідок.

Вид головної форми. На головній формі розміщені три кнопки, призначення яких відповідає написам на кнопках згідно рисунку 5.1.

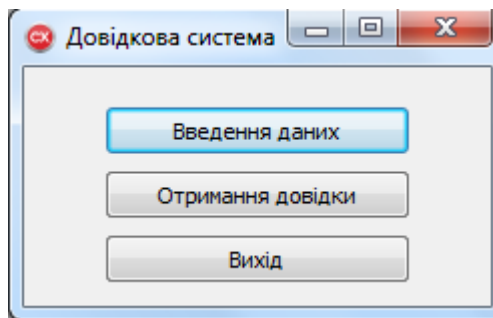


Рис.5.1 – Вигляд вікна головної форми

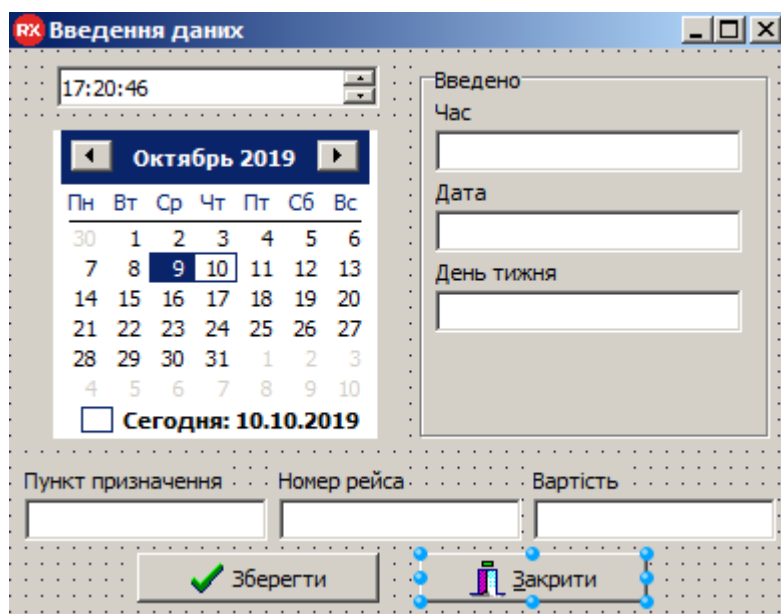
Крім головної форми проект містить ще дві форми, одна з яких призначена для введення даних, а інша — для отримання довідки.

При розробці вікна введення інформації для введення дати і часу потрібно використовувати компоненти MonthCalendar і DateTimePicker, а також Button або BitBtn, Label, Edit та LabeledEdit.

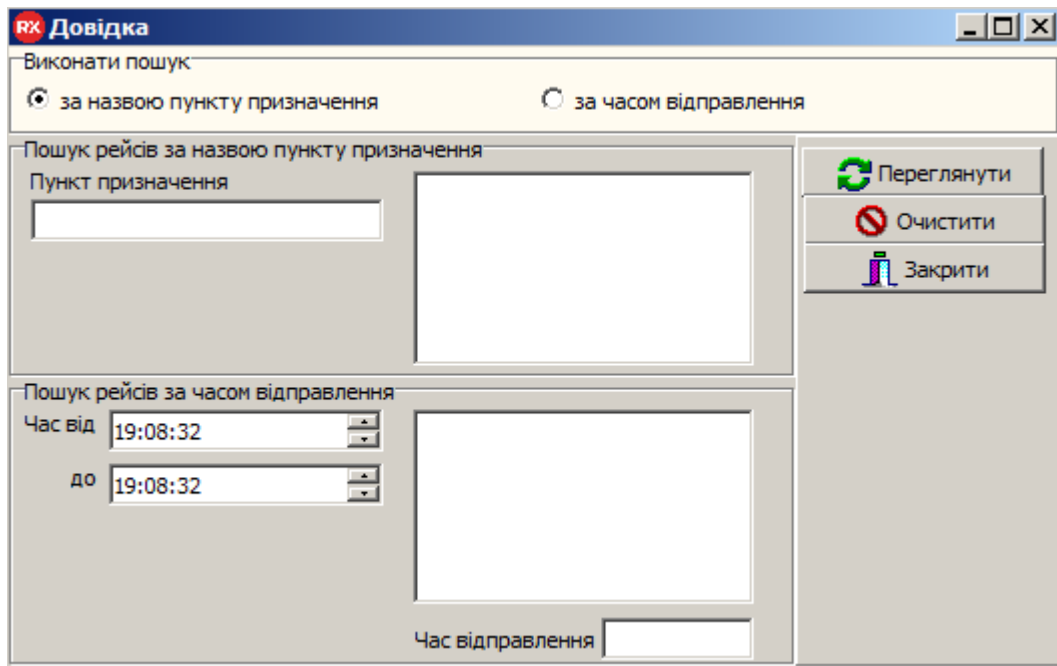
Для компоненту DateTimePicker встановити властивість Kind як dtkTime.

Компоненти MonthCalendar і DateTimePicker призначені для введення дати та часу відправлення літаків. У полях, розміщених унизу форми, користувач вводить відповідну вхідну інформацію. Введені дані розміщуються в запис масиву після натискання на кнопку «Зберегти». Одночасно з заповненням масиву записів на панелі «Введено» підтверджується введена інформація. Після введення кількості записів, яка відповідає встановленому обмеженню, виводиться відповідне повідомлення і введення даних припиняється.

Зразок розташування компонентів показаний на рисунку.



Вікно видачі довідок містить дві панелі (компоненти GroupBox), призначення яких слідує з написів. На кожній панелі розміщені по дві кнопки та по одному компоненту ListBox, в які виводиться довідкова інформація. Крім того, на панелях розміщуються компоненти Label, Edit, LabeledEdit.



Є також кнопка для закриття вікна.

Додати до проекту також модуль без форми, для цього виконати команду головного меню File->New->Unit. Модуль зберегти у проекті під ім'ям UnCom, в інші форми підключати в uses.

У модуль UnCom внести оголошення глобальних типів та змінних, так щоб файл містив, наприклад, наступне:

```
#include <System.Classes.hpp>
class tmas {
public:
    AnsiString Punkt;
    int nom_rejsa;
    double st_bil;
    TDateTime wremja;
    tmas() { Punkt="-"; nom_rejsa=0; st_bil=0; wremja=NULL; }
};
int ogr_mas(3);
int kol_mas(0);

String A1[7] = {" Субота ", " Неділя ", " Понеділок ",
    " Вівторок ", " Середа ", " Четвер ", " П'ятниця "};
```

Нижче наведено частину програмного коду для обробників подій модулів.

Підключаємо за необхідністю модулі Unit2.h, Unit3.h, UnCom.h в #include.

Обробник виклику другої форми:

```
Form2->ShowModal();
```

Обробник закриття:

```
Close();
```

```
//-----
```

```
// Файл Unit2.pas
```

```
Обробник введення даних:
```

```
void __fastcall TForm2::Button1Click(TObject *Sender)
```

```
{
```

```
if (BitBtn1->Tag > ogr_mas-1) {
```

```
    ShowMessage("Не можна більше ввести ");
```

```
    exit;
```

```
};
```

```
LabeledEdit1->Text = TimeToStr(DateTimePicker1->Time);
```

```
LabeledEdit2->Text = DateToStr(MonthCalendar1->Date);
```

```
LabeledEdit3->Text = A1[DayOfWeek(MonthCalendar1->Date)];
```

```
Form3->mas[kol_mas].Punkt = LabeledEdit4->Text;
```

```
Form3->mas[kol_mas].nom_rejsa = StrToInt(LabeledEdit5->Text);
```

```
Form3->mas[kol_mas].st_bil = StrToFloat(LabeledEdit6->Text);
```

```
//double fraction, integer;
```

```
    /*mas[kol_mas].wremja = DateTimePicker1->Time;
```

```
//ShowMessage('Время = ' + TimeToStr( mas[kol_mas].wremja));
```

```
    BitBtn1->Tag = BitBtn1->Tag + 1;
```

```
    kol_mas = kol_mas + 1;
```

```
// ShowMessage( 'kol_mas '+ IntToStr( kol_mas));
```

```
// ShowMessage('Button1.Tag = ' + IntToStr( Button1.Tag));
```

```
}
```

```
//-----
```

```
-----
```

```
void __fastcall TForm2::Button2Click(TObject *Sender)
```

```
{
```

```
Close();
```

```
}
```

```
//-----
```

```
-----
```

// Файл Unit3.cpp

//-----

Обробник отримання довідки:

```
class TForm3 : public TForm
{
...
public:          // User declarations
    __fastcall TForm3(TComponent* Owner);
    tmas mas[3];
};

void __fastcall TForm3::BitBtn1Click(TObject *Sender)
{
    int i;
    if (RadioGroup1->ItemIndex==0) {
        for (i=0; i<ogr_mas; i++) {
            ShowMessage(mas[i].Punkt);
            if (mas[i].Punkt == LabeledEdit1->Text) {
                ListBox1->Items->Add(IntToStr(mas[i].nom_rejsa));
            }
        }
    }
    else {

        String s;
        TDateTime ogr1, ogr2;
        for (i=0; i<ogr_mas; i++) {
            ogr1 = DateTimePicker1->Time;
            ogr2 = DateTimePicker2->Time;
            if ((mas[i].wremja < ogr2) && (mas[i].wremja > ogr1)) {
                ListBox2->Items->Add(mas[i].Punkt);
                LabeledEdit2->Text = TimeToStr(mas[i].wremja);
            }
        }
    }
}
//-----
-----
void __fastcall TForm3::BitBtn2Click(TObject *Sender)
{
    if (RadioGroup1->ItemIndex==0) { ListBox1->Clear(); } else {
        ListBox2->Clear(); }
}
//-----
```

Обробник «Очистити»:

```
ListBox1->Clear();
```

Обробник «Закрити»:

```
Close();
```

Оформити звіт, описавши проєктування.

ЛАБОРАТОРНА РОБОТА № 11-13

Розробка та тестування багатовіконної програми з використанням компонентів з вкладками та діалогів. Використання компонентів меню

Мета: вивчення прийомів розробки та тестування багатовіконної програми з використанням компонентів з вкладками та діалогів, використанням компонентів меню

Постановка задачі. Доповнити програму, розроблену в лабораторній роботі 10, що вводить інформацію про рейси літаків і надає довідку, головним і контекстним меню, використанням багатосторінкового блокноту для отримання довідки, можливістю збереження довідки в файл.

Порядок виконання роботи наведений нижче.

Головну форму «Довідкова система» змінити таким чином, щоб вона не містила кнопок, а містила компонент MainMenu (Standard).

Подвійним щигликом на компоненті MainMenu1 викликати редактор меню (Menu Designer). Для завдання основних пунктів меню (лінійки меню) необхідно переміщуватись вправо по горизонталі, активізуючи в міру необхідності чергове поле для розташування в ньому пункту лінійки меню. Щоб створити для пункту, розташованого в лінійці меню, списку елементів підменю, що розкривається, потрібно для цього пункту активізувати в міру необхідності чергове поле, що спускається вниз.

Створити пункти меню "Введення даних", «Отримання довідки», «Вихід».

Для створення пункту меню " Введення даних" потрібно увійти в Object Inspector, задати заголовок (властивість Caption) пункту - " Введення даних" і ім'я пункту (властивість Name) - N1.

Оброблювати властивість OnClick аналогічно натисканню кнопки.

Задати обробку виклику кожного пункту меню.

Додати на форму «Отримання довідки» компонент PageControl. Натиснути над ним праву клавішу миші і вибрати “New Page”. Аналогічно додати другу сторінку.

Використовуючи властивість Caption, задати сторінкам назви «Довідка щодо номерів рейсів», «Довідка щодо пунктів призначення». Перенести на ці сторінки відповідні GroupBox.

Додати на кожну сторінку в вікно довідок можливість збереження даних з ListBox в файл, використовуючи компонент SaveDialog. Запит на збереження викликати за допомогою компонента PopupMenu.

Всі візуальні компоненти мають властивість PopupMenu. Якщо на компоненті клацнути правою кнопкою миші, виникає відповідне цього компоненту меню. Перш ніж підключати спливаюче меню, його треба створити.

Для появи спливаючого меню треба встановити його властивість AutoPopup в True. Властивість Alignment визначає, де саме з'явиться це меню.

Процес розробки спливаючого меню аналогічний створенню головного меню.

Спливаюче меню обробляє події OnPopup безпосередньо перед своєю появою на екрані.

Для відкриття вікна Menu Designer двічі клацнути на спливаючому меню.

Помістіть в це меню пункт «Зберегти» .

Для компонента ListBox привласнити як значення властивості PopupMenu ім'я компонента PopupMenu1 .

Для SaveDialog1 знайти в Інспекторі Об'єктів властивість Filter, зробити щиглик по кнопці із трьома крапками й у редакторі фільтрів, що відкрився, внести в поле FilterName:

текстові файли (*.txt)

у поле Filter:

*.txt

Для компонента SaveDialog1 знайти в Інспекторі Об'єктів властивість DefaultExt і ввести в нього txt.

Оформити звіт, описавши проєктування.

ЛАБОРАТОРНА РОБОТА № 14-15

Створення та відлагодження програми з підключенням до бази даних з використанням візуальних та невізуальних компонентів

Мета: вивчення прийомів створення та відлагодження програми з підключенням до бази даних з використанням візуальних та невізуальних компонентів

Основні теоретичні відомості

Для розробки структури таблиць та їх записів можна використовувати спеціальні програми, які містять засоби для створення таблиць, зміни їх структури і редагування їх записів. Дії з управління структурою таблиць можна виконати і програмно. При проектуванні враховуються правила нормалізації, використання первинних та зовнішніх ключів.

Частина 1

Постановка завдання

Створити БД (допускається довільна СКБД, рекомендується MySQL), яка містить таблиці «Група» (ключ, назва групи), «Студент» (ключ, прізвище, група (зовнішній ключ), номер залікової книжки). Створити проєкт, в якому можна переглядати список груп та студентів в групі, виконувати додавання, редагування та видалення даних про студентів, виконувати фільтрацію та сортування. Серед даних повинна бути інформація з власним прізвищем та групою, яку навести в індивідуально оформленому звіті.

Хід роботи

Створити БД з двома вказаними таблицями.

Навести структурну схему взаємозв'язку таблиць.

Заповнити таблицю груп.

Розробити програмний додаток, що працює з базою даних і виконує такі функції:

- відображення бази даних;
- виконання навігації;
- модифікація даних (додавання, редагування, видалення);
- фільтрація за назвою групи.

На формі при виконанні має бути надпис з власним прізвищем.

При створенні нового додатку передбачити модуль даних (DataModule), в якому рекомендовано розташувати компоненти з вкладки ADO (або іншої) палітри компонентів:

- ADOConnection, ADOTable;
- DataSource.

Навести на рисунку вигляд модуля даних.

Форма має містити DBGrid, DBNavigator, DBLookupComboBox (для фільтрації по групі) або аналогічні компоненти.

Рекомендовано встановити значення властивостей:

Об'єкт	Властивість	Значення
ADOConnection1	ConnectionString	Use Connection String – Build –Вказати джерело даних
	LoginPrompt	False
ADOTable1	Connection	ADOConnection1
	TableName	Назва таблиці
DataSource1	DataSet	Вибрати зі списку ADOTable1
DBGrid	DataSource	Вибрати зі списку DataSource
DBNavigator		
DBLookupComboBox1	DataSource (окремий)	ListSource (джерело з переліком груп), ListField (назва групи) та KeyField (ID)
Button1...	Caption	Очистити фільтр

Передбачити обробку подій OnCreate, OnDestroy для DataModule1, в яких задати відповідно підключення та відкриття джерела даних та закриття.

Виконати дії перегляду, додавання записів, видалення записів, редагування записів, фільтрації та сортування.

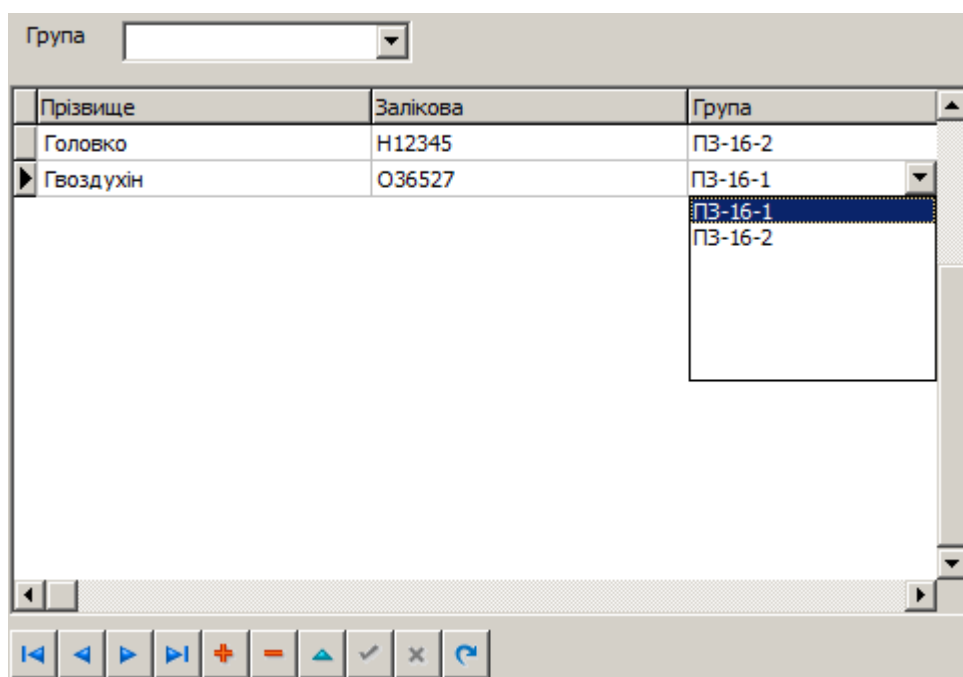
Передбачити фільтрацію за вибраною групою в DBLookupComboBox1

```
DataModule2->ADOTable1->Filtered=false;
```

```
DataModule2->ADOTable1->Filter="(ID_GROUP="+... /*ключ групи*/
...+")";
```

```
DataModule2->ADOTable1->Filtered=true;
```

Зразок фрагменту вікна наведений нижче.



Продемонструвати роботу програми, оформити звіт, прокоментувати індивідуальний хід роботи та виконане самостійно проектування (описати використані властивості, методи, події тощо).

Частина 2

Постановка завдання. Доповнити функціонал використанням компонентів ADOQuery на заміну ADOTable як такими, що забезпечують більш гнучкий механізм роботи з наборами даних. В таблиці перегляду приховати стовпці, що відповідають полям з первинними ключами.

Передбачити скидання фільтру за назвою групи, наприклад, за натисканням кнопки).

Додати в таблицю Студент поле з назвою файлу фотографії студента.

Передбачити на формі справа від DBGrid панель для виведення зображення.

Зображення рекомендовано виводити в Image (або DBImage).

Продемонструвати роботу програми, оформити звіт, прокоментувати індивідуальний хід роботи та виконане самостійно проектування (описати використані властивості, методи, події тощо).

ЛАБОРАТОРНА РОБОТА № 16-17

Реалізація сортування даних. Контроль даних при вводі у поля таблиці

Мета: вивчення прийомів реалізації сортування даних та контролю даних при вводі у поля таблиці

Частини 1-2

Постановка завдання

Доповнити функціонал програми лабораторної роботи 15 наведеним нижче.

Передбачити сортування даних за кліком по стовпцю DBGrid за зразком, наведеним нижче або з використанням властивостей ADOQuery. Допускається використання ORDER BY в SQL-запиті. Приклад використання обробника DBGrid1TitleClick для сортування по стовпцю (Column) за відповідним до цього стовпця полем наведений в лістингу 16.1

Лістинг 16.1 - Сортування засобами ADOTable1

```
DataModule2->ADOTable1->Close();  
DataModule2->ADOTable1->IndexFieldNames = /*назва поля*/;  
DataModule2->ADOTable1->Open();
```

Доповнити функціонал модифікацією запису в окремій модальній формі, яка викликається з контекстного меню над DBGrid і заборонити модифікацію в DBGrid. Модальна форма повинна мати властивість Name=Form##, де ## - двозначний номер за варіантом і Caption=Prizvyshe, де Prizvyshe - прізвище студента, що виконує лабораторну роботу. Передбачити коригування всіх полів, окрім первинного ключа. Для виведення та редагування полів рекомендовано використати компонент ADOQuery.

Таким чином, модальна форма має забезпечити:

- модифікацію даних за вказаними нижче варіантами;
- введення інформації щодо прізвища студента, номера залікової книжки та фото студента, вибір групи для студента зі списку - за варіантами.

Додати перевірку і попередження, якщо вже є інший запис про студента з даним прізвищем та ініціалами.

Пояснити проектування в звіті, навести скріншот виконання форми з індивідуальним зображенням (будь-яка картинка замість фото студента).

Варіанти:

- непарні — можливість додавання нового запису в таблицю "Студент"; вибір групи зі списку (аналог ListBox або DBGrid);

- парні — можливість редагування поточного запису в таблиці "Студент"; вибір групи з випадаючого списку (аналог ComboBox або DBLookupComboBox).

Продемонструвати роботу програми, оформити звіт, прокоментувати індивідуальний хід роботи та виконане самостійно проектування (описати використані властивості, методи, події тощо).

ЛАБОРАТОРНА РОБОТА № 18-19

Розробка та відлагодження програми динамічного формування запитів до БД

Мета: вивчення прийомів розробки та відлагодження програми динамічного формування запитів до БД

Частина 1

Постановка завдання

Розробити програму, яка працює з БД "Аеропорт", до складу полів таблиці якої входять номер рейсу, назва пункту призначення, дата вильоту, вартість проданих білетів (дійсне), кількість проданих білетів.

Внести в заголовок форми власне прізвище.

Реалізувати в програмі додавання та модифікацію поточного запису. При редагуванні запису контролювати, щоб номер рейсу знаходився у заданих межах. Назву пункту призначення обирати зі заздалегідь сформованого списку назв пунктів призначення. При введенні нового запису відстежувати, щоб номер рейсу не повторювався у БД.

Заборонити додавання та модифікацію даних в таблиці перегляду (DBGrid). Для додавання та модифікації використовуються дані, введені у відповідній панелі внизу. Для введення дати використати відповідно налаштовані компоненти.

Кнопка «Вихід» завершує роботу програми.

Для кнопок додавання та модифікації у властивість Hint внести текст «Введіть дані у всі поля панелі вводу» та встановити ShowHint = true.

Підготувати перелік назв пунктів призначення, який повинен бути введений у випадаючий список, та запрограмувати його завантаження перед початком роботи програми.

Зразок розташування елементів інтерфейсу програми показане на рисунку. В правій частині вікна розташовані компоненти DBEdit. Надписи стовпців таблиці перегляду виконати українською.

Nom_rejs	Punkt	Vremja_v	Vart_bil	Date_v
12	Dnepr	15:10:00	34,5	18.03.2010
24	Київ	10:00:00	25,9	15.04.2010
34	Ташкент	15:41:42	400,55	18.11.2002
50	Тбілісі	16:44:42	533	18.11.2002

Рисунок

Обробник натискання на кнопку «Вихід» Button7Click:

```
Close();
```

Обробник натискання на кнопку «Вставити» Button11Click (Locate виконує пошук):

```
int j, Nom_Tek_Zap;
Nom_Tek_Zap = Table4->RecNo;
Table4->First();
if (Table4->Locate('Nom_rejs', Edit4->Text, TLocateOptions
())<<loPartialKey<<loCaseInsensitive) {
    ShowMessage('Номер рейса вже є!');
    exit;
}
Table4->RecNo = Nom_Tek_Zap;
Table4->Insert();
Table4->FieldByName('Nom_rejs')->Value = StrToInt(Edit4.Text);
Table4->FieldByName('Vart_bil')->Value = StrToFloat(Edit5.Text);
Table4->FieldByName('Punkt')->Value = Edit6.Text;
Table4->FieldByName('Date_v')->Value = DateTimePicker2.Date;
Table4->Post();
```

Для реалізації завдання можна використати компоненти ADOQuery.

Частина 2

Постановка завдання

Реалізувати пошук рейсів із заданою датою вильоту та рейсів, номери яких належать заданому інтервалу. Здійснювати сортування за датою вильоту, вартістю білету.

Елементи інтерфейсу програми, призначені для пошуку у БД рейсів із заданою датою виліту розміщені на панелі з заголовком: «Знайти рейс на дату» і керуються кнопкою з написом «Запит по датам».

Елементи інтерфейсу програми, призначені для пошуку у БД рейсів із заданого інтервалу розміщені на панелі з заголовком: «Рейси з інтервалу» і керуються кнопкою з написом «Показати рейси з інтервалу». Для введення обмежень на номери рейсів використовуються компоненти CSpinEdit або подібні.

Крім того, для відновлення вигляду БД після запитів, на формі розміщена кнопка «Показати всю БД».

Протестувати програму і переконатися в правильності її роботи.

Пояснити використання TQuery замість TTable або використати їх в програмі.

ЛАБОРАТОРНА РОБОТА № 20-21

Виведення підсумкових даних в статусний рядок. Використання конфігураційного файлу для підключення до БД

Мета: закріплення навичок виведення підсумкових даних в статусний рядок та використання конфігураційного файлу для підключення до БД

Теоретична частина.

Компонент `StatusBar` складається з набору панелей типу `TStatusPanels`, що відтворюють рядок стану. Розташовується, як правило, внизу форми.

Ini-файл — це текстовий файл з розширенням `.ini`, призначений для зберігання налаштувань програми. Текст в такому файлі згрупований по розділам, наприклад:

```
[Розділ1]  
Ідентифікатор1=Значення1
```

Клас `TIniFile` оголошений в `IniFiles`. Можна читати значення за допомогою різних методів, таких як, наприклад, `ReadString`. Зворотній метод для запису значення — `WriteString`.

Частина 1

Постановка завдання

Взяти за основу БД з таблицями "Групи" і "Студенти", розроблену в одній з попередніх лабораторних робіт і розроблену програму з фільтрацією по групах, заповнити умовно для студентів груп "ПЗ-19-1", "ПЗ-19-2", "ПЗ-20п-1", вивести дані в `DBGrid`. Вивести підсумкові дані за варіантом в статусний рядок. Відлагодити програму. Оформити звіт, у скріншоті виконання навести фрагмент вікна з можливістю перевірки.

Варіант 1. Вивести в статусний рядок кількість студентів групи "ПЗ-19-1".

Варіант 2. Вивести в статусний рядок кількість студентів обраної у фільтрі групи, у яких номер залікової книжки містить цифру 7.

Варіант 3. Вивести в статусний рядок кількість студентів групи "ПЗ-19-1", чиє прізвище містить підрядок "ко".

Варіант 4. Вивести в статусний рядок кількість груп.

Варіант 5. Вивести в статусний рядок кількість груп, у студентів яких номер залікової книжки містить цифру 0.

Варіант 6. Вивести в статусний рядок кількість студентів всіх груп, у яких прізвище закінчується на "ко".

Варіант 7. Вивести в статусний рядок кількість студентів групи "ПЗ-19-2".

Варіант 8. Вивести в статусний рядок кількість груп, у студентів яких прізвище закінчується на "ко".

Варіант 9. Вивести в статусний рядок кількість студентів групи "ПЗ-19-2", чиє прізвище містить підрядок "ов".

Варіант 10. Вивести в статусний рядок кількість груп, де є студенти з прізвищами, що починаються з літер "А", "Б", "В".

Варіант 11. Вивести в статусний рядок кількість студентів групи "ПЗ-19-1", чиє прізвище містить підрядок "нко" або "нько".

Варіант 12. Вивести в статусний рядок кількість студентів обраної у фільтрі групи, у яких номер залікової книжки починається з 1.

Варіант 13. Вивести в статусний рядок кількість студентів групи "ПЗ-20п-1".

Варіант 14. Вивести в статусний рядок кількість студентів обраної за фільтром групи, у яких поле з номером залікової книжки складається з 5 цифр або символів.

Варіант 15. Вивести в статусний рядок кількість груп, де є студенти з прізвищами, що починаються з літер "Г", "Д", "К".

Варіант 16. Вивести в статусний рядок кількість студентів обраної за фільтром групи, у яких не заповнене поле з номером залікової книжки.

Варіант 17. Вивести в статусний рядок кількість студентів групи "ПЗ-20п-1", чиє прізвище містить підрядок "ін".

Варіант 18. Вивести в статусний рядок кількість студентів обраної за фільтром групи.

Варіант 19. Вивести в статусний рядок кількість студентів обраної у фільтрі групи, у яких номер залікової книжки починається з "123".

Варіант 20. Вивести в статусний рядок кількість груп, де є студенти з прізвищами, що починаються з літер "Л", "М", "С".

Частина 2

Постановка завдання

Передбачити по натисканню на кнопку збереження в конфігураційному файлі з назвою Prizvyshe.ini, де Prizvyshe – прізвище студента, налаштувань для підключення до БД або інших налаштувань. Секція повинна мати назву і значення, вказані за варіантом. По натисканню на іншу кнопку передбачити виведення налаштувань на форму з можливістю модифікації і збереження в конфігураційному файлі. Відлагодити програму. Оформити звіт.

Варіант 1. Секція "DB", значення "Host", "Port".

Варіант 2. Секція "DBConnect", значення "Username", "Password".

Варіант 3. Секція "DBConfig", значення "DriverName", "LoginPrompt".

Варіант 4. Секція "Database", значення "DriverName", "DBName".

Варіант 5. Секція "DBRemote", значення "RemoteHost", "Port".

Варіант 6. Секція "DBBackup", значення "BackupTime", "BackupDir".

Варіант 7. Секція "Database", значення "RootName", "RootPassword".

Варіант 8. Секція "Database", значення "ServerName", "DBName".

Варіант 9. Секція "DB", значення "AutoCommit", "DataDir".

Варіант 10. Секція "Database", значення "DateFormat", "TimeFormat".

Варіант 11. Секція "Database", значення "SQLLogOff", "SQLWarnings".

Варіант 12. Секція "DBConfig", значення "LogError", "LogWarnings".

Варіант 13. Секція "DBConfig", значення "License", "ProtocolVersion".

Варіант 14. Секція "DBConfig", значення "MaxUserConnections", "MaxJoinSize".

Варіант 15. Секція "Database", значення "LogOutput", "LogError".

Варіант 16. Секція "DBConfig", значення "DataDir", "DateTimeFormat".

Варіант 17. Секція "DBConfig", значення "CollationDatabase", "CollationServer".

Варіант 18. Секція "DB", значення "CharacterSetDatabase", "CharacterSetServer".

Варіант 19. Секція "Database", значення "AutoCommit", "Log".

Варіант 20. Секція "Database", значення "License", "BaseDir".

ЛАБОРАТОРНА РОБОТА № 22-23

Створення та відлагодження програми побудови графіку на основі набору даних

Мета: вивчення прийомів створення та відлагодження програми побудови графіку на основі набору даних

Теоретична частина

Компонент Chart дозволяє будувати графіки та діаграми і має свої властивості, методи та події. Компонент є контейнером об'єктів Series типу TChartSeries — серій даних, що характеризуються різними стилями відображення. Кожний компонент Chart може включати декілька серій. Кожна серія відповідає одній кривій на графіку. Подібним компонентом, пов'язаним безпосередньо з джерелом даних, є DBChart.

Для побудови графіку на основі набору даних додати на форму компонент TChart. Виконати подвійний клік над об'єктом графіку і у віконці обрати Series Add, як наведено на рисунку. Обрати вид графіку (лінійний, гістограма), зняти позначку "3D".

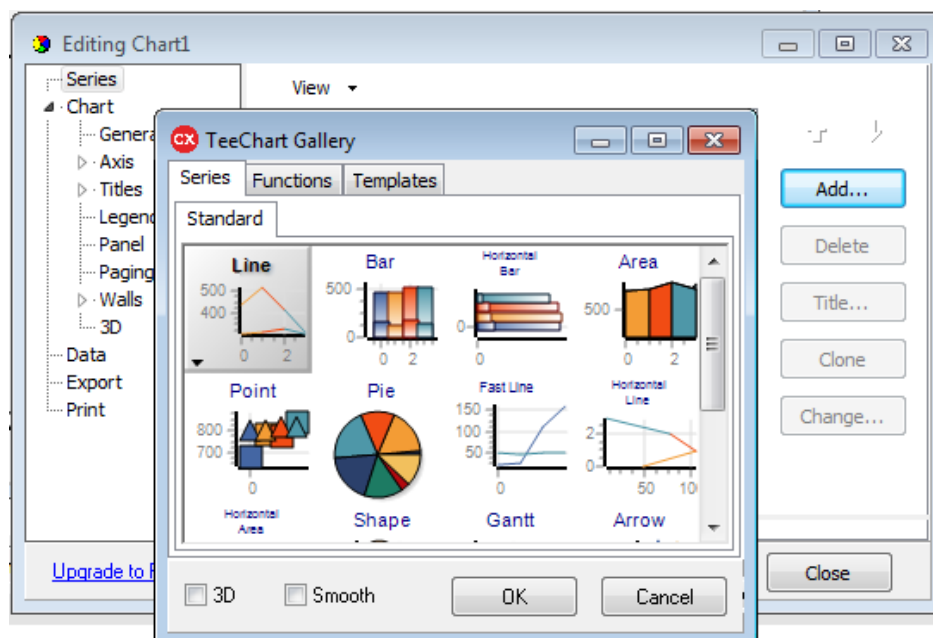


Рис. – Вибір виду графіку

В обробнику необхідної події (клік по пункту меню, клік на кнопку, створення форми тощо) на основі логічного набору даних, SQL-запиту з

урахуванням фільтрів побудувати серію точок на графіку за прикладом, наведеним нижче. Зразок вигляду графіку наведений на рисунку.

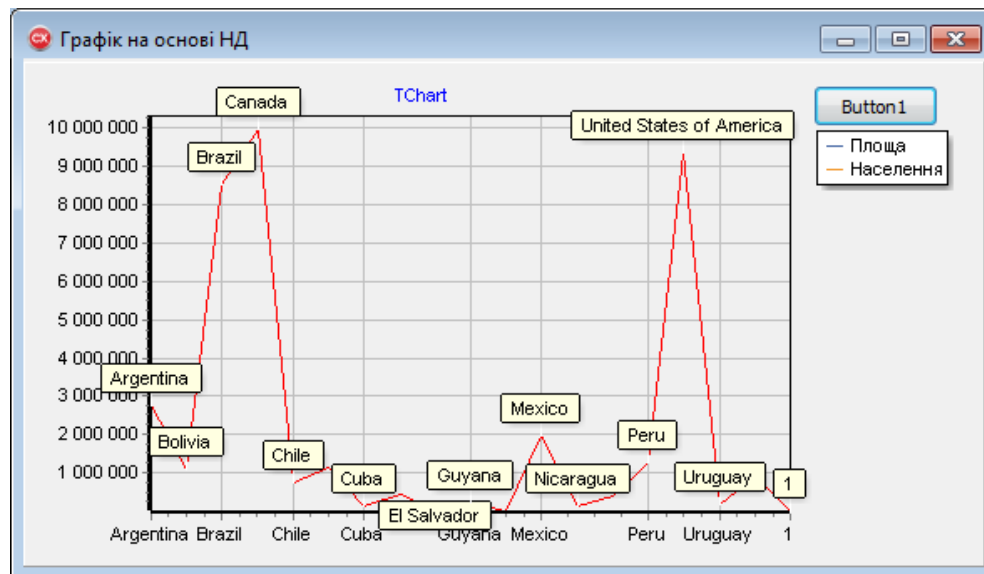


Рис. – Зразок вигляду графіку

Лістинг – Побудова серії точок на графіку

```
Series1->Clear(); //Очистити серію
Chart1->Title->Text->Clear();
Chart1->Title->Text->Add("My Title"); //Заголовок
//Series1->Marks->Visible=True;
DataModule1->Query1->Close();
DataModule1->Query1->SQL->Text="SELECT  name,  capital,  continent,
area, population FROM country";
DataModule1->Query1->Open();
DataModule1->Query1->First();
int i(0);
while (!(DataModule1->Query1->Eof)) {
    float area=DataModule1->Query1->FieldByName("area")->AsFloat;
    float
        population=DataModule1->Query1-
>FieldByName("population")->AsFloat;
    String s=DataModule1->Query1->FieldByName("name")->AsString;
    i++;
    Series1->AddXY(i,area,s,clRed);
    DataModule3->Query1->Next();
}
```

Постановка завдання

Взяти за основу БД з таблицями "Групи" і "Студенти", розроблену в одній з попередніх лабораторних робіт і розроблену програму з фільтрацією по групах, додати в таблицю "Студенти" поля "Рейтинг за осінній семестр" та "Рейтинг за весняний семестр", заповнити ретинговими балами для студентів груп "ПЗ-19-1", "ПЗ-19-2", "ПЗ-20п-1", вивести дані в DBGrid. Рейтинг вважати за 100-бальною

системою. По натисканню на кнопку "Графік" побудувати на окремій формі графік на основі даних за варіантом. Доповнити графік заголовком з власним прізвищем. Відлагодити програму. Оформити звіт, у скріншоті виконання навести фрагмент вікна з даними і побудованим графіком.

Варіант 1. Графік рейтингу за осінній семестр для студентів групи "ПЗ-19-1".

Варіант 2. Графік рейтингу за весняний семестр для студентів групи "ПЗ-20п-1".

Варіант 3. Графік по групах з середнім балом рейтингу для кожної групи за осінній семестр.

Варіант 4. Графік по групах з кількістю студентів, які мають рейтинг від 90 до 100.

Варіант 5. Графік з середнім балом рейтингу за осінній та весняний семестр для студентів групи "ПЗ-19-1".

Варіант 6. Графік з сумарним балом рейтингу за осінній та весняний семестр для студентів групи "ПЗ-19-1".

Варіант 7. Графік рейтингу за осінній семестр для студентів групи "ПЗ-19-2".

Варіант 8. Графік рейтингу за весняний семестр для студентів групи "ПЗ-19-1".

Варіант 9. Графік з сумарним балом рейтингу за осінній та весняний семестр для студентів групи "ПЗ-19-2".

Варіант 10. Графік по групах з кількістю студентів, які мають рейтинг від 74 до 89.

Варіант 11. Графік з середнім балом рейтингу за осінній та весняний семестр для студентів групи "ПЗ-19-2".

Варіант 12. Графік з сумарним балом рейтингу за осінній та весняний семестр для студентів групи "ПЗ-20п-1".

Варіант 13. Графік рейтингу за осінній семестр для студентів групи "ПЗ-20п-1".

Варіант 14. Графік рейтингу за весняний семестр для студентів групи "ПЗ-19-2".

Варіант 15. Графік по групах з середнім балом рейтингу для кожної групи за весняний семестр.

Варіант 16. Графік по групах з максимальним балом рейтингу за осінній семестр.

Варіант 17. Графік по групах з кількістю студентів, які мають рейтинг від 60 до 73.

Варіант 18. Графік з середнім балом рейтингу за осінній та весняний семестр для студентів групи "ПЗ-20п-1".

Варіант 19. Графік по групах з максимальним балом рейтингу за весняний семестр.

Варіант 20. Графік по групах з кількістю студентів, які мають рейтинг від 0 до 59 включно.

ЛАБОРАТОРНА РОБОТА № 24-25

Створення та відлагодження програми побудови графіків

Мета: вивчення прийомів створення та відлагодження програми побудови графіків

Постановка задачі. Задається функція $y=f(x)$, потрібно побудувати графік цієї функції. Інтервал значень незалежної змінної x задається в програмі.

Теоретичні відомості.

За поверхню, на яку програма може здійснювати виведення графіки, відповідає об'єкт `Canvas`. Властивість `Pixels`, що представляє собою двовимірний масив типу `TColor`, містить інформацію про колір кожної точки графічної поверхні. Використовуючи властивість `Pixels`, можна задати колір будь-якої точки графічної поверхні, тобто "намалювати" точку. Наприклад, інструкція

```
Canvas->Pixels[10, 10] := clRed
```

зафарбовує точку поверхні форми в червоний колір.

Розмірність масиву `Pixels` визначається реальним розміром графічної поверхні. Розмір графічної поверхні форми (робочої області, що також називають клієнтською) визначають властивості `ClientWidth` і `ClientHeight`, а розмір графічної поверхні компонента `Image` — властивості `Width` і `Height`. Лівий верхній точці робочої області форми відповідає елемент `Pixels[0, 0]`, а правий нижній — `Pixels[ClientWidth - 1][ ClientHeight -1]`.

Наступна програма, використовуючи властивість `Pixels`, будує графік функції $y = 2 \sin(x) \exp(x/5)$. Межі діапазону зміни аргументу функції є вихідними даними. Діапазон зміни значення функції обчислюється під час роботи програми. На підставі цих даних програма обчислює масштаб, що дозволяє побудувати графік таким чином, щоб він займав всю область форми, призначену для виведення графіка. Для побудови графіка використовується вся доступна область форми, причому якщо під час роботи програми користувач змінить розмір вікна, то графік буде виведений заново з урахуванням реальних розмірів вікна.

Порядок виконання.

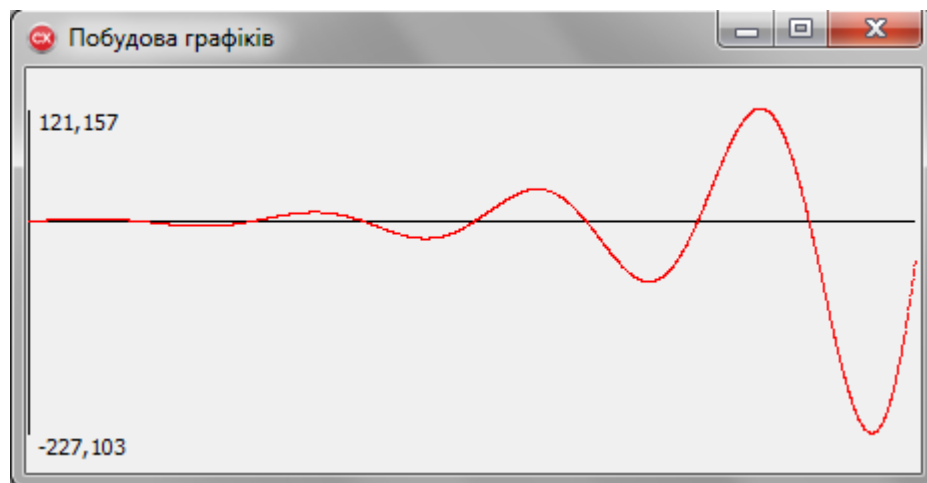
Відкрити новий додаток. У текст файлу специфікації форми внести текст з оголошенням процедури побудови Grafik

```
class TForm1 : public TForm {
__published: // IDE-managed Components
private: // User declarations
    void Grafik();
public: // User declarations
    __fastcall TForm1(TComponent* Owner);
};
```

У текст файлу реалізації модуля внести такі зміни:

- розрахунок $f(x)$ оформити у вигляді окремої функції;
- описати реалізацію функції Grafik(); обробників *OnPaint*, *OnResize*.

Текст модуля приведено.



Лістинг модуля

```
//-----
...
//-----
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    Grafik();
}
//-----
-----
void __fastcall TForm1::FormResize(TObject *Sender)
{
    TRect rct;
    rct = Rect(0,0, ClientWidth, ClientHeight);
    Canvas->FillRect(rct); //стерти
    Grafik();
}
```

```

}

double f(double x)
{
    double y;
    y=2*sin(x)*exp(x/5);
    return y;
}

void TForm1::Grafik()
{
    double x, y, x1(0), y1, x2(25), y2, dx(0.01), mx, my;    //
    int l(10), b, w, h, x0, y0;    //
    b = Form1->ClientHeight-20;    //
    h = Form1->ClientHeight-40;    //
    w = Form1->Width - 20;    //
    x = x1;
    y1 = f(x);    //
        y2 = f(x);    //
        x = x+dx;
    do {
        y = f(x) ;
        if ( y < y1) { y1 = y; }
        if ( y > y2) { y2 = y; }
        x = x+dx;
    } while (x < x2);
    my = h/fabs(y2-y1);    //
    mx = w/fabs(x2-x1);    //
    x0 = SimpleRoundTo(1+fabs(x1*mx), 0);
    y0 = SimpleRoundTo(b-fabs(y1*my), 0);
    Canvas->MoveTo(x0,b);    Canvas->LineTo(x0,b-h);
    Canvas->MoveTo(l,y0);    Canvas->LineTo(l+w,y0);
    Canvas->TextOut(x0+5,b-h,FloatToStrF(y2,ffGeneral,6,3));
    Canvas->TextOut(x0+5,b,FloatToStrF(y1,ffGeneral, 6,3));
    x = x1;
    do {
        y = f(x);
        Canvas->Pixels[SimpleRoundTo(x0+x*mx,0)][SimpleRoundTo(y0-
y*my,0)] = clRed;
        x = x+dx;
    } while (x < x2);
}

//-----

```

Основну роботу виконує функція Grafik (її оголошення треба помістити в розділ private оголошення форми в заголовному файлі програми). Функція Grafik спочатку обчислює максимальне (y_2) і мінімальне (y_1) значення функції на відрізьку $[x_1, x_2]$. Потім, використовуючи інформацію про ширину й висоту області виводу графіка, вона обчислює коефіцієнти масштабування по осях X і Y. Після цього обчислює координату Y горизонтальної осі, координату X

вертикальної осі й вичерчує координатні осі. Потім виконується безпосередня побудова графіка.

Виклик функції `Grafik` виконують функції обробки подій `OnPaint` і `OnResize`. Процедура `TForm1.FormPaint` забезпечує креслення графіка після появи форми на екрані у результаті запуску програми а також після перекриття вікна програми іншими вікнами.

Процедура `TForm1.FormResize` забезпечує креслення графіка після зміни розмірів форми.

Доповнити програму згідно завдання за варіантом:

Варіант 1.

Вивести на екран графік функції $y=2*\sin(x)*\exp(x/10)$ синім кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 2.

Вивести на екран графік функції $y=2*\cos(x)*\exp(x/5)$ зеленим кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 3.

Вивести на екран графік функції $y=3*\sin(x)*\exp(x/2)$ синім кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 4.

Вивести на екран графік функції $y=0.5*\cos(x)*\exp(x/3)$ зеленим кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 5.

Вивести на екран графік функції $y=3.5*\sin(x)*\exp(x/4)$ синім кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 6.

Вивести на екран графік функції $y=5*\cos(x)*\exp(x/5)$ зеленим кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 7.

Вивести на екран графік функції $y=0.2*\sin(x)*\exp(x/5)$ синім кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 8.

Вивести на екран графік функції $y=0.6*\cos(x)*\exp(x/4)$ червоним кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 9.

Вивести на екран графік функції $y=2.2*\sin(x)*\exp(x/2)$ червоним кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 10.

Вивести на екран графік функції $y=3.6*\cos(x)*\exp(x/3)$ зеленим кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 11.

Вивести на екран графік функції $y=2*\sin(x)*\exp(x/10)$ синім кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 12.

Вивести на екран графік функції $y=2*\cos(x)*\exp(x/5)$ зеленим кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 13.

Вивести на екран графік функції $y=3*\sin(x)*\exp(x/2)$ синім кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 14.

Вивести на екран графік функції $y=0.5*\cos(x)*\exp(x/3)$ зеленим кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 15.

Вивести на екран графік функції $y=3.5*\sin(x)*\exp(x/4)$ синім кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 16.

Вивести на екран графік функції $y=5*\cos(x)*\exp(x/5)$ зеленим кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 17.

Вивести на екран графік функції $y=0.2*\sin(x)*\exp(x/5)$ синім кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 18.

Вивести на екран графік функції $y=0.8*\cos(x)*\exp(x/6)$ червоним кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 19.

Вивести на екран графік функції $y=2.2*\sin(x)*\exp(x/2)$ червоним кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 20.

Вивести на екран графік функції $y=3.6*\cos(x)*\exp(x/3)$ зеленим кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 21.

Вивести на екран графік функції $y=5*\cos(x)*\exp(x/5)$ зеленим кольором, підписати на графіку точку локального максимуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 22.

Вивести на екран графік функції $y=0.2*\sin(x)*\exp(x)$ білим кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 23.

Вивести на екран графік функції $y=3*\cos(x)*\exp(x)$ червоним кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 24.

Вивести на екран графік функції $y=2.2*\sin(x)*\exp(x/2)$ червоним кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Варіант 25.

Вивести на екран графік функції $y=2*\cos(x)*\exp(x/3)$ чорним кольором, підписати на графіку точку локального мінімуму у вигляді (x, y) , замість (x, y) написати фактичні значення аргумента та функції.

Виконати програму, прокоментувати, оформити звіт.

ЛАБОРАТОРНА РОБОТА № 26-27

Створення та відлагодження багатовіконного MDI-додатку

Мета роботи: вивчення прийомів створення та відлагодження багатовіконного MDI-додатку

Постановка задачі.

Розробити програму з інтерфейсом множини документів — простий багатовіконний додаток.

Теоретичні відомості.

Основним елементом будь-якої програми є форма — контейнер, в якому розміщуються інші візуальні і невізуальні компоненти. З точки зору користувача форма — це вікно, в якому він працює з додатком. Кожній новій формі, введеній в додаток, відповідає свій модуль (unit), що описує цю форму як клас і включає, якщо необхідно, якісь додаткові константи, змінні, функції і процедури.

В додатках MDI дочірні форми не можуть бути автоматично створюваними, так як число таких форм визначає користувач під час роботи програми, створюючи кожен нову форму командою типу «Нове вікно».

Для кожної автоматично створюваної форми додається у файл програми відповідний оператор її створення методом `CreateForm`. Це можна побачити, якщо виконати команду `Project | View Source` і переглянути файл проекту. Він може, наприклад, містити наступні виконувані оператори:

```
Application->CreateForm(__classid(TForm1), &Form1);  
Application->CreateForm(__classid(TForm2), &Form2);  
Application->Run();
```

Перший і другий з них створюють відповідні форми, а третій починає виконання програми.

Для форм, які були виключені зі списку автоматично створюваних, аналогічний метод `CreateForm` треба виконати в той момент, коли форма повинна бути створена.

У момент створення форми виникає подія OnCreate. Обробка цієї події широко використовується для налагодження якихось компонентів форми, створення списків і т.ін.

Для створення програми MDI необхідно спроектувати батьківську і дочірню форми. У батьківській формі властивість FormStyle встановлюється в fsMDIForm, а в дочірній — в fsMDIChild. Оскільки дочірні вікна буде створювати сам користувач в процесі виконання програми, дочірню форму необхідно виключити з числа створюваних автоматично за допомогою вікна опцій проекту.

Розробка інтерфейсу

Почнемо з побудови форми вікна документа.

1. Створіть новий додаток.

2. Назвіть форму, що відкрилася FDoc.

3. Помістіть на формі компонент RichEdit1 типу TRichEdit. Його властивість Align зробіть рівним alClient, щоб вікно редагування зайняло всю площу вікна. Зітріть текст, що з'явився в RichEdit1 (властивість Lines).

4. Змініть властивість форми FormStyle на fsMDIChild.

5. Збережіть проект, давши спроектованому модулю ім'я, наприклад, UDoc.

6. Спроектуємо батьківську форму. Відкрийте у вашому проекті нову форму (File | New Form). Назвіть її FMDI. Збережіть її модуль з ім'ям, наприклад, UMDI (File | Save As).

7. Змініть властивість FormStyle нової форми на fsMDIForm. Можете задати властивість WindowState рівним wsMaximized, щоб вікно цієї форми пред'являлося користувачеві в перший момент розгорнутим на весь екран. В include модуля UMDI введіть посилання на модуль дочірньої форми UDoc.

8. Виконайте команду Project | Options і у вікні на сторінці Forms переведіть дочірню форму FDoc зі списку автоматично створюваних в список доступних. При цьому головною стане батьківська форма FMDI — єдина, що залишилася в списку автоматично створюваних форм.

9. Помістіть на формі список зображень ImageList і диспетчер дій ActionList. Пошліться у властивості Images компонента ActionList1 на ImageList1.

10. В редакторі ActionList1 (відкрити подвійним кліком) за натисканням правої клавіші миші введіть дію NewAction, яке буде відповідати створенню нового вікна документа, впишіть власне прізвище. Оброблювач її події OnExecute може мати вигляд:

```
void __fastcall TFMDI::Action1Execute(TObject *Sender) {
    TFDoc *NewF=new TFDoc(Application);
    NewF->Caption="Документ      Іванова      І.І.      "+IntToStr(FMDI-
>MDIChildCount);
    NewF->Show();
}
```

11. Призначте цей же обробник у події форми OnShow, щоб у перший момент відкривалося одне вікно документа.

12. У диспетчері дій ActionList введіть стандартні дії розділу Window: WindowCascade, WindowTileHorizontal, WindowTileVertical. Напишіть обробники їх подій OnExecute. Для дії «Каскад»: Cascade(); Для дії «Впорядкувати по горизонталі»: TileMode=tbHorizontal; Tile(); Для дії «Впорядкувати по вертикалі»: TileMode=tbVertical; Tile();

Введіть на форму компонент MainMenu. Сформуйте в ньому розділ меню "Вікно" і пошліться в його підрозділах на введені вами дії.

На цьому етап проектування багатовіконного додатку завершений.

Додайте до RichEdit1 контекстне меню, за допомогою якого напишіть завантаження тексту з текстового файлу.

Оформіть звіт, прокоментуйте програму.

ЛАБОРАТОРНА РОБОТА № 28

Створення власного компонента. Проектування додатку з використанням власного компонента. Відлагодження та тестування програми

Мета: вивчення прийомів проектування і створення користувачем власного компонента. Вивчення прийомів проектування додатку з використанням власного компонента, відлагодження та тестування програми

Постановка завдання. У середовищі C++ Builder розробити власний компонент користувача, протестувати його.

Частина 1. Створення модуля компонента

Перед початком роботи зі створення нового компонента потрібно створити спочатку каталог для модуля і інших файлів компонента. Після цього можна приступити до створення компонента.

Щоб почати роботу над новим компонентом, треба активізувати процес створення нового елемента (команда File | New | Other), в меню Individual Files вибрати Component і VCL For C++ Win32, вказати базовий клас TEdit для створюваного компонента (рис. 1).

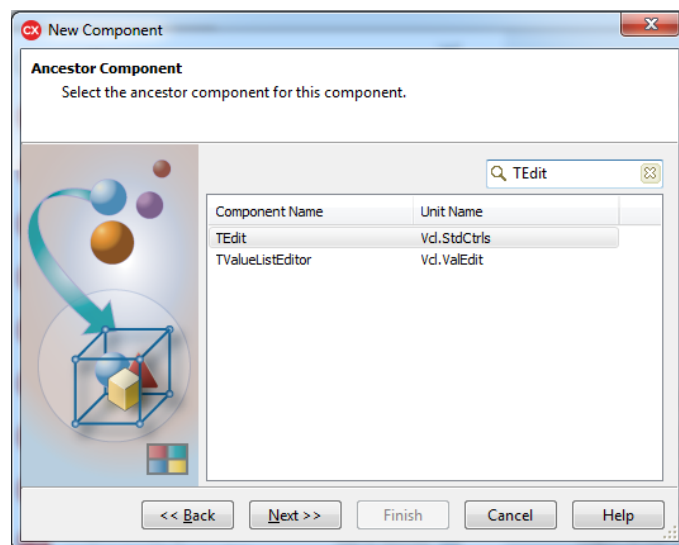


Рис. 1. Початок роботи над новим компонентом

У полі Ancestor type треба ввести базовий тип створюваного компонента. Для компоненту базовим компонентом є стандартний компонент Edit (поле введення-редагування). Тому базовим типом компоненту, що розглядається, є тип TEdit.

У полі Class Name необхідно ввести ім'я класу компоненту, який розробляється (виконати за варіантом TNew##Edit, де ## – двозначний номер за списком).

У полі Palette Page потрібно ввести ім'я вкладки палітри компонентів, на яку буде поміщений значок компонента. Назва вкладки палітри компонентів можна вибрати із списку. Якщо в полі Palette Page ввести ім'я ще не існуючої вкладки палітри компонентів, то безпосередньо перед додаванням компонента вкладка з вказаним ім'ям буде створена.

У полі Unit name знаходиться автоматично сформоване ім'я файлу модуля створюваного компонента. Клацнувши на кнопці з трьома крапками, вибрати і навести в звіті каталог, в якому повинен бути збережений модуль компонента.

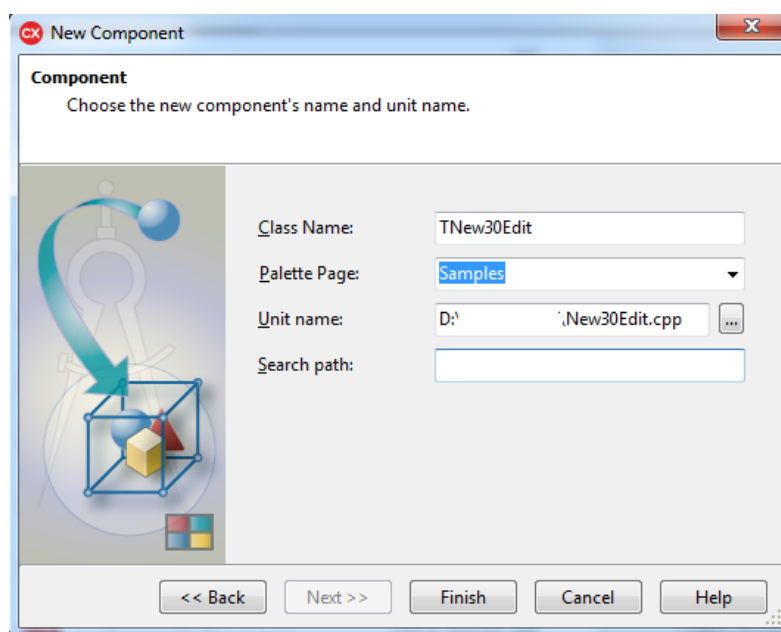


Рис. 1. Інформація для нового компонента

В результаті клацання на кнопці ОК буде сформований модуль компонента New##Edit (в кожного свій номер).

У файлі знаходиться оголошення нового класу та розміщена функція Register, яка забезпечує реєстрацію, установку значка компонента на вказану вкладку палітри компонентів.

У сформований середовищем шаблон компонента можна внести доповнення: оголосити поля даних, функції доступу до полів даних, властивості та методи. Якщо на деякі події компонент повинен реагувати не так, як базовий, то в оголошення класу потрібно помістити оголошення відповідних функцій обробки подій (перевизначити).