МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ІНЖЕНЕРНО-ХІМІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА «АВТОМАТИЗАЦІЯ ХІМІЧНИХ ВИРОБНИЦТВ»

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ З КРЕДИТНОГО МОДУЛЯ « ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ ПРИКЛАДНИХ ПРОГРАМ » КУРСУ “ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ – 2”

для студентів напрямку підготовки “6.050202 – Автоматизація та комп’ютерно-інтегровані технології ”

Київ
НТУУ «КПІ»
2014

Методичні вказівки до лабораторних робіт з кредитного модуля «Візуальне програмування прикладних програм» курсу “Прикладне програмне забезпечення – 2” для студентів напрямку підготовки “ 6.050202- Автоматизація та комп’ютерно-інтегровані технології ” [Текст] / Укладач, В. М. Ковалевський, // – К.: НТУУ «КПІ», 2014. – 53 с.

Гриф надано Методичною і Вченою радою ІХФ «КПІ»

(Протокол № 5 від 23 червня 2014 р.)

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ З КРЕДИТНОГО МОДУЛЯ « ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ ПРИКЛАДНИХ ПРОГРАМ » КУРСУ “ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ – 2”

для студентів напрямку підготовки “6.050202 – Автоматизація та комп’ютерно-інтегровані технології ”

Укладач: *Ковалевський Валерій Михайлович,* канд. техн. наук, доцент.

Відповідальний
за випуск: *А. І. Жученко,* док. техн. наук, професор.

Рецензент: *О. М. Тимонін,* канд. техн. наук, доцент.

Авторська редакція

Зміст

стор.

В С Т У П

Лабораторні роботи з візуального програмування прикладних програм орієнтовані на вивчення техніки та методик по розробкам і побудуванню в ***C***++ ***Builder*** прикладних програм на мовах програмування ***C*** та ***C***++.

На протязі навчального семестру з кредитного модуля «Візуальне програмування прикладних програм» курсу « Прикладне програмне забезпечення - 2» лабораторні роботи виконуються відповідно до графіку з лабораторних занять, який наведено у таблиці В-1.

Графік проведення лабораторних занять, кількості навчальних годин та тем і змісту занять.

Таблиця В-1.

№ п/п заняття	Кількість годин заняття	Зміст і тема заняття
1	2	Інструктаж бакалаврів по таких питаннях: - техніки безпеки при роботи по виконанню лабораторних робіт; - особливості візуального програмування в інтегрованому програмувальному середовищі C++ Builder; - методика і правила використання редактора коду C++ Builder для розробки листингів при візуальному програмуванні прикладних програм; - вимоги по підготовки бакалаврів до занять та по оформленню результатів до захисту виконаних завдань по лабораторним роботам.
2	2	Виконання вказівок і команд по протоколу лабораторної роботи № 1 на тему «Імітаційне моделювання відображення інформації на дисплеях мікропроцесорного приладу ІТМ-11».
3	2	Виконання вказівок і команд по протоколу лабораторної роботи № 2 на тему «Техніка зчитування ScanCode та ASCII кодів клавіш клавіатури та їх використання прикладних C++ програмах».
4	1	Захист створеної прикладної програми у лабораторній роботі № 1 та результатів внесення змін у програму відповідно до індивідуального завдання.
	1	Захист створеної прикладної програми у лабораторній роботі № 2 та результатів внесення змін у програму відповідно до індивідуального завдання.

5	2	Виконання вказівок і команд по протоколу лабораторної роботи № 3 на тему «Техніка використання технології Drag & Doc і плаваючих вікон в прикладних С++ програмах».
6	2	Виконання вказівок і команд по протоколу лабораторної роботи № 4 на тему «Алгоритми з розробки і використання фреймів у С++ Builder при створенні прикладної програми».
7	1	Захист створеної прикладної програми у лабораторній роботі № 3 та результатів внесення змін у програму відповідно до індивідуального завдання.
	1	Захист створеної прикладної програми у лабораторній роботі № 4 та результатів внесення змін у програму відповідно до індивідуального завдання.
8	2	Виконання вказівок і команд по протоколу лабораторної роботи № 5 на тему «Компоненти С++ Builder для візуалізації структури ієрархічних даних на носіях інформації».
9	1	Захист створеної прикладної програми у лабораторній роботі № 5 та результатів внесення змін у програму відповідно до індивідуального завдання.
	1	Оформлення звіту та нарахування балів до виконаних і захищених лабораторних робіт кредитного модуля.

Лабораторна робота № 1

Імітаційне моделювання відображення інформації на дисплеях мікропроцесорного приладу ІТМ-11

Ціль виконання лабораторної роботи. Передбачається виконання таких навчальних задач:

- розробка прикладної програми яка буде імітувати вивід числових значень на дисплеях у зображенні панелі мікропроцесорного приладу;

- вивчення алгоритму по створенню комп'ютерної імітаційної моделі з роботи мікропроцесорного приладу ІТМ-11, використовуючи стандартні компоненти інтегрованого програмувального середовища ***C++ Builder***.

Відомості до приладу ІТМ-11

У технологічних процесах для показу значень технологічних параметрів використовуються прилади ІТМ-11(рис. 1.1). Ці прилади обладнані також функцією сигналізації та інтерфейсом зв’язку RS-485.

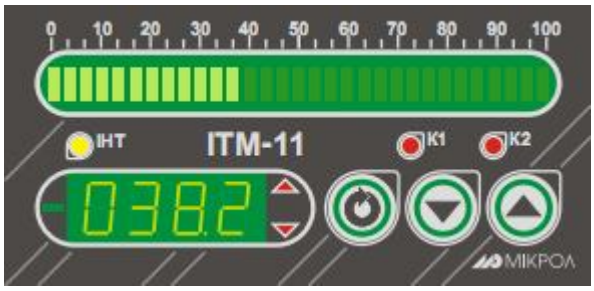

Рис 1.1. Передня панель ІТМ-11.

Індикатор технологічний мікропроцесорний ІТМ-11:

- Універсальні багатофункціональні та одноканальні індикатори значень технологічних параметрів;
- Модифікації ІТМ-11 - горизонтальне виконання з цифровим дисплеєм і лінійним сегментним індикатором (31 сегмент).

Область застосування:

- Системи цифрового і лінійної індикації технологічних параметрів;
-Двопозиційний, трипозиційний і багатопозиційне регулювання температури, тиску, витрат, рівня й інших фізичних величин;
- Дистанційні пристрої зв'язку з об'єктом керування;
- Територіально розподілені і локальні системи управління;
- Місцеві щити та пульти керування, мнемосхеми і т.п.

Функціональні можливості:

- Робота з уніфікованими сигналами, термоперетворювачами опору, термопарами;
- Кожен аналоговий вхід може бути налаштований на підключення будь-якого типу датчика;

- Індикація параметрів в технологічних одиницях на цифрових і лінійних (0-100%) індикаторах;
- Цифрове калібрування (автоматичне і ручне) початку шкали і діапазону вимірювання;
- Вибір методу лінійної індикації (сегмент, гістограма);
- Завдання та сигналізація відхилення від установок мінімум і максимум на передній панелі приладу;
- Тип технологічної сигналізації: без запам'ятовування спрацьовування, із запам'ятовуванням спрацьовування і квітируванням;
- Вхідний цифровий фільтр аналогових входів;
- Витяг квадратного кореня (вимірювання витрати по перепаду тиску);
- Функція вимірювання інтегральної витрати;
- Лінеаризація вхідного сигналу (по 16 точкам):
- Конфігуровані дискретни виходи - транзистор, реле, оптосімістор або твердотільне реле;
- Програмована логіка роботи вихідних пристроїв: більше (***MAX***), менше (***MIN***), в зоні (***MIN-MAX***), поза зоною **(*MIN-MAX*)**;
- Аналоговий вихід для ретрансміссії вхідних аналогових сигналів;
- Архівація даних в енергонезалежну пам'ять (планується модернізація);
- Збереження параметрів при відключенні живлення;
- Захист від несанкціонованої зміни налаштованих параметрів;
- Гальванічно розділений інтерфейс ***RS***-485, протокол ***ModBus RTU*** (збір інформації, конфігурація).

Постановка задачі по програмуванню у лабораторній роботі

Необхідно створити ***C***++ програму для роботи у ***Windows*** і у який буде вікно з зображенням панелі приладу ІТМ-11 горизонтального виконання, цифровим дисплеєм та лінійним сегментним індикатором.

Для створення імітаційної прикладної програми на формі необхідно використовувати таки компоненти ***C++ Builder*** :

Image – компонента із вкладки ***Additional*** для завантаження рисунків і відображення на формі;

Timer – компонента із вкладки ***System***, необхідна для створення анімованого зображення;

Label – компонента із вкладки ***Standard*** , необхідна для відображення тексту.

TrackBar – компонента із вкладки ***Win***32, необхідна для завдання параметру.

Edit - компонента із вкладки ***Standard***, для забезпечення введення мінімального та максимального значення контрольованого параметру.

Chart - компонента із вкладки ***Additional*** для графічного відображення значення контрольованого параметру.

Завдання для аналізу і обробки результатів роботи запрограмованої C++ програми

Після запуску в роботу ***C++*** програми лабораторної роботи № 1 для захисту результатів програмування необхідно виконати таке:

- нарисувати блок-схему алгоритму з роботи програми лабораторної роботи № 1;
- нарисувати блок-схему алгоритму по виконанню операторів функцій, з обробки у програмі подій.

Порядок дій і команд для виконання програмування задачі:

Крок 1. Створюємо новий проект ***C++***, перейменовуємо назву форми проекту через властивість ***Caption***. Змінюємо назву ***Form*1** на Емулятор ІТМ-11. Зберігаємо зміни. Папку з картинками із назвою ІТМ-11 поміщаємо в ту ж папку де знаходяться файли збереженого проекту.

Крок 2. Розміщуємо на формі програми необхідні компоненти:

Timer1 – Interval=100; Timer2 - Interval=100; Timer3 - Interval=100.

Image1,Image2, Image3, Image4, Image5, Image6, Image7, Image8.(Для кожного з Image підбираємо необхідні розміри і позиції, усі зміни параметрів робимо через Object Inspector - Properties).

Image1 – AutoSize – true; Picture – завантажуєм ITM-11.bmp; Left =0;Top=0;

Image2 - Left =62; Top=88; Height=28; Width=20; Picture – завантажуємо ITM-11\Digits\0.bmp

Image3 - Left =84; Top=88; Height=28; Width=20; Picture –завантажуємо ITM-11\Digits\0d.bmp

Image4 - Left =106; Top=88; Height=28; Width=20; Picture –завантажуємо ITM-11\Digits\0.bmp

Image5 – Stretch=true; Left =22;Top=28; Height=23; Width=251; Picture – завантажуємо ITM-11\Linear\0.bmp

Image6 – Transparent=true; Left =133; Top=106; Height=17; Width=17; Picture – завантажуємо ITM-11\at.bmp

Image7 – Transparent=true; Left =133; Top=87; Height=17; Width=17; Picture – завантажуємо ITM-11\at2.bmp

Image8 - Left =38; Top=88; Height=28; Width=20; Picture –завантажуємо ITM-11\Digits\0.bmp

TrackBar1 – Min=0, Max=100; SelStart=30; SelEnd=70;

Label1 – змінюємо параметр Font, настроюємо бажаний шрифт і розмір шрифту.

Label2 – змінюємо параметр Caption на Контроль параметру

Розміщуємо на формі компоненти ***Edit*1** та ***Edit*2** і змінюємо параметри для них:

Edit1 – Text – min;

Edit2 – Text – max.

Додаємо компоненту ***Chart***, бачимо на формі область для побудови графіку, двічі клікаємо по ній ЛКМ, обираємо ***Add***… із вкладки ***Series***,

з'являється ***Series*1**. Переходимо на вкладку ***Titles*** і називаємо графік «Контрольований параметр»

Після того як усі компоненти розставлені, змінюємо властивості форми Form1 – AutoSize = true;

Форма з усіма необхідними компонентами повинна мати вигляд, як показано на рисунку 1.2.

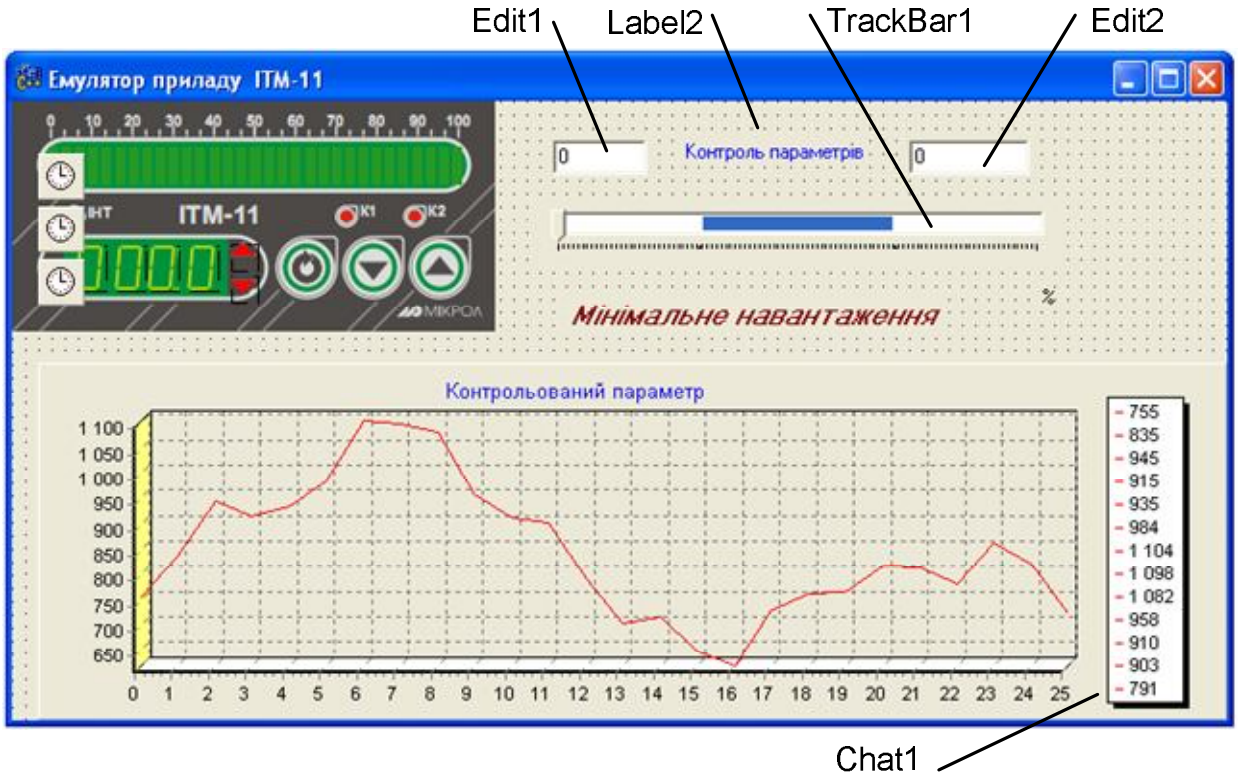

Рис. 1.2. Форма вікна програми із розміщеними на ній компонентами.

<u>Крок 3.</u> Заповнюємо лістинг програми, створюючи необхідні події і додаючи вказівки препроцесорів. Усі події створюємо або подвійним кліком мишки по компоненті, або через ***Object Inspector – Events***.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <math.h>
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
float k,P[100],x1,ch;
int x,c,c2,c3,c4,i,min=0,max=10,i2,j,l;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
 Form1->DoubleBuffered=true;
 Image6->Visible=false;
 Image7->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
   k=(max-min)*0.01;
   x1=min+(TrackBar1->Position)*k;
   x=min+(TrackBar1->Position)*k;
   c=(x-x%100)/100;                    //перша цифра числа
   c2=((x-x%10)-(x-x%100))/10;  //друга цифра
   c3=x%10;                             //третя цифра
   ch=ceil((x1-x)*10);
   c4=ch;                    //цифра після коми
   l=(TrackBar1->Position*31)/100; //відповідає за заповнення сегментів
   if (max<10)
  {Image8->Picture->LoadFromFile("ITM-11/Digits/0.bmp") ;
   Image2->Picture->LoadFromFile("ITM-11/Digits/0.bmp") ;
   Image3->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c3)+"d.bmp") ;
   Image4->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c4)+".bmp");
   Image5->Picture->LoadFromFile("ITM-11/Linear/"+IntToStr(l)+".bmp");
  }
  if (max<=99 )
 {
  Image8->Picture->LoadFromFile("ITM-11/Digits/0.bmp") ;
  Image2->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c2)+".bmp") ;
  Image3->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c3)+"d.bmp") ;
  Image4->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c4)+".bmp");
  Image5->Picture->LoadFromFile("ITM-11/Linear/"+IntToStr(l)+".bmp");
  }
    if (max<1000 )
 {
  Image8->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c)+".bmp") ;
  Image2->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c2)+".bmp") ;
  Image3->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c3)+"d.bmp") ;
  Image4->Picture->LoadFromFile("ITM-11/Digits/"+IntToStr(c4)+".bmp");
  Image5->Picture->LoadFromFile("ITM-11/Linear/"+IntToStr(l)+".bmp");
}
 if (TrackBar1->Position<=30)
 { Timer2->Enabled=true;Timer3->Enabled=false; Image7->Visible=false;
```

```
  Label1->Caption="  Мінімальне навантаження ";
  Label1->Font->Color=clGreen;}
 if (TrackBar1->Position>=70)
 {Timer3->Enabled=true;Timer2->Enabled=false; Image6->Visible=false;
  Label1->Caption="  Максимальне навантаження ";
  Label1->Font->Color=clRed;}
 if (TrackBar1->Position>30 && TrackBar1->Position<70)
 {i=0;
  Label1->Caption="  Робочий режим ";
  Image6->Visible=false;
  Image7->Visible=false;
  Timer2->Enabled=false;
  Timer3->Enabled=false;
  Label1->Font->Color=clBlue;}
  //*** Рисування графіка функції***//
   P[100]= x+ch/10;
   for (i2=0;i2<100;i2++)
  {P[i2]=P[i2+1];}
   Series1->Clear();
   for (j=0;j<100;j++)
   {
     Series1->AddY(P[j]);
   }
 }
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
  i++;
  if (i==1)
  {Image6->Visible=true;}
  if(i==2)
  {Image6->Visible=false;i=0;}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer3Timer(TObject *Sender)
{
  i++;
  if (i==1)
  {Image7->Visible=true;}
  if(i==2)
  {Image7->Visible=false;i=0;}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
 min=StrToInt(Edit1->Text);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Edit2Change(TObject *Sender)
{
 max=StrToInt(Edit2->Text);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2Click(TObject *Sender)
{
 Edit2->Text=IntToStr(max);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Click(TObject *Sender)
{
Edit1->Text=IntToStr(min);
}
//---------------------------------------------------------------------------
```

Готова робоча програма лабораторної роботи повинна мати вікно, як показано на рисунку 1.3.

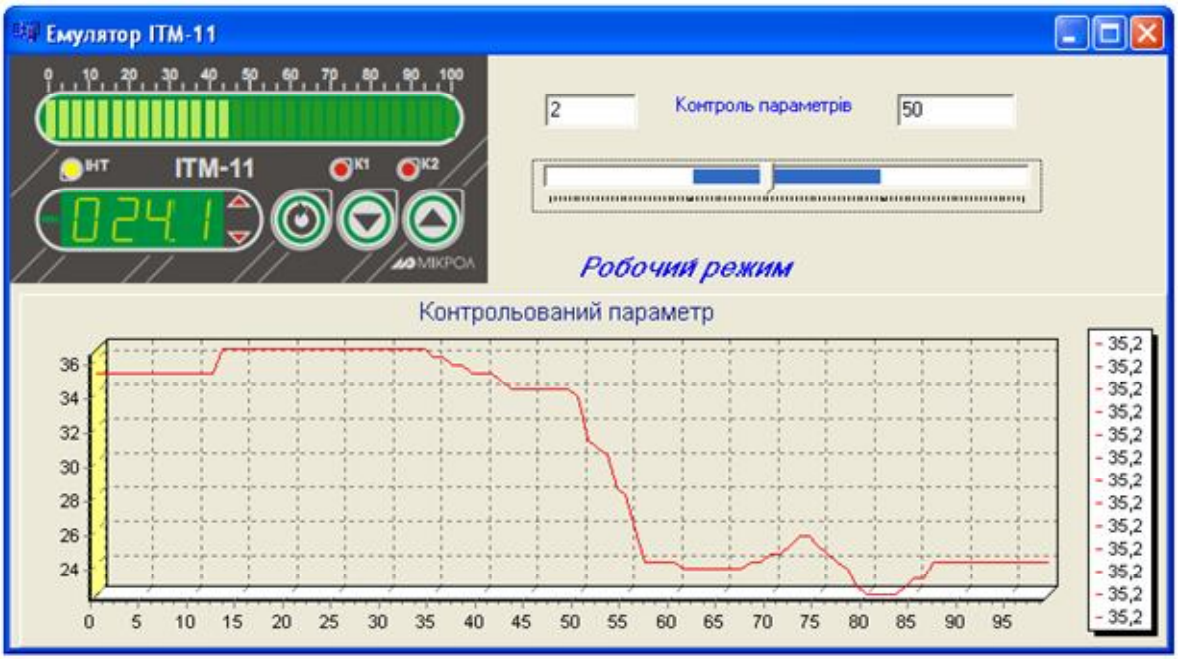

Рис. 1.3. Зображення вікна програма з демонстрації роботи дисплеїв приладу ІТМ-11.

Контрольні запитання до лабораторної роботи № 1

1. Пояснить загальні правила активізації (ініціалізації) основних файлів проекту, створюваної програми у ***C++ Builder***.

2. Яки команди на блок схемі алгоритму виконуються для обробки події до появи на полі вікна графіка.
3. Пояснить алгоритм та початкові тексти ***C***++ програми по обробці події з показу числових значень на полі цифрового дисплея.
4. Пояснить алгоритм та початкові тексти ***C***++ програми по обробці події до лінійного сегментного індикатора.
5. Покажіть команди ***C***++ ***Builder***, якими виконується налаштування компонент для обробки подій мишки.

Лабораторна робота № 2

Техніка зчитування ScanCode та ASCII кодів клавіш клавіатури та їх використання в прикладних C++ програмах

Ціль виконання лабораторної роботи. Передбачено виконання таких навчальних задач:

- вивчення ***ASCII*** кодів до символів та їх використання у прикладних програмах при виконанні в операційному середовищі ***MS DOS***;
- практичне вивчення техніки використання ASCII кодів до символів для побудови графічних зображень.

⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇

Загальні зведення до *ScanASCII Code*

Кожен символ, зазначений на кнопці клавіатури, характеризується стандартним ASCII кодом, який можна використовувати в C++ програмах. Наприклад, для перевірки умови, клавіша з яким символом була натиснута, необхідно записати в умову ***ASCII*** код відповідного символу. У стандартних ***ASCII*** таблицях також маються коди для псевдографічних значків, за допомогою яких можна в текстовому режимі роботи прикладної програми на екрані будувати різні різні рамки та інші фігури і позначення. На клавіатурі

кожна кнопка позначається відповідним ***Scan*** кодом, що разом з ***ASCII*** кодом символу формує в буфері клавіатури ***Scan-ASCII*** код. При натисканні клавіші із символом на клавіатурі в буфер клавіатури записується определенный Scan-ASCII Code, по якому апаратні переривання ***BIOS*** операційної системи ПК визначають, що введено з клавіатури в С++ програму.

Постановка задачі по програмуванню у лабораторній роботі

У лабораторній роботі № 2 необхідно виконати наступні дві задачі:

Задача А. Необхідно створити за допомогою ***Consol Wizard*** файл ***C***++ програми, що виконується, у вигляді консольного додатка ***MS DOS***. Дана прикладна програма повинна по діагоналі вікна будуватися три рамки за допомогою псевдографічних значків. У цих рамках потрібно забезпечити можливість уведення будь-яких символів із клавіатури і на границі рамки для останнього введеного символу на екрані потрібно показати ASCII код останнього символу.

Задача Б. Потрібно створити за допомогою ***Consol Wizard*** файл ***C***++ програми, що виконується, у вигляді консольного додатка ***MS DOS*** При роботі С++ програми на екрані повинні з'являтися запити на введення координат лівого верхнього кута рамки і координат правого нижнього кута рамки. Після введення координат кутів рамки на екрані повинні з'явитися кутові псевдографічні значки, а при натисканні будь-якої клавіші клавіатури (пробіл) на екрані повинні малюватися лінії рамки.

Завдання для аналізу і обробки результатів роботи запрограмованої С++ програми

Після запуску в роботу ***C***++ програм до задач **А** та **Б** з лабораторної роботи № 2 для захисту результатів програмування необхідно виконати таке:

- нарисувати блок-схему алгоритму до прикладної програми з проекту ***P_Lab2_A*** лабораторної роботи № 2;
- нарисувати блок-схему алгоритму до прикладної програми з проекту ***P_Lab2_Б*** лабораторної роботи № 2.

⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇

Методика виконання задачі А.

***Порядок дій і команд при виконанні програмування задачі А*:**

Крок 1. За допомогою кнопки ***New*** відкрийте вікно ***New Items*** і виберіть ***Console Wizard*** і для опції ***Source Type*** задайте режим ***C***++. В результаті ***C***++ ***Builder*** відкриє шаблон тексту вихідної прикладної програми в такому вигляді:

```
//---------------------------------------------------------------------------
#pragma hdrstop
//---------------------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
        return 0;
}
//---------------------------------------------------------------------------
```

Крок 2. Додайте у відповідні місця шаблону прикладної С++ програми вказівки препроцесора, оголошення і оператори у відповідності з наступним текстом програми

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <conio.h>
#include <system.hpp>
#include <stdio.h>
#define dx   24     /* Ширіна рамки вікна */
#define dy   5      /* Висота рамки вікна */
void my_box(int xul,int yul,int xlr,int ylr,int btype)
  {
static int boxcar[2][6] = { /* Графічни символи для контурів */
 { 218,196,191,179,192,217 },     /* одинарний контур     */
 { 201,205,187,186,200,188 }      /* двійний контур       */
  };
    int i,hzchar,vtchar;
```

```
   if(btype) {
      hzchar = boxcar[btype -1][1];
      vtchar = boxcar[btype -1][3];
 /* Вивід верхньої та ніжньої сторони контуру   */
 gotoxy(xul,yul);
   for(i = xul; i<= xlr; i++) putch(hzchar);
 gotoxy(xul,ylr);
   for(i = xul; i<= xlr; i++) putch(hzchar);
/* ввід правоі та лівоі сторони контуру  */
   for(i = yul; i<= ylr; i++)
     { gotoxy(xul,i); putch(vtchar);
       gotoxy(xlr,i); putch(vtchar);
     }
/* Вивід кутів контуру */
/*  верхнього лівого кута */
gotoxy(xul,yul); putch( boxcar[btype - 1][0] );
/* верхнього правого кута */
gotoxy(xlr,yul); putch( boxcar[btype - 1][2] );
/* ніжнього лівого кута */
gotoxy(xlr,ylr); putch( boxcar[btype - 1][5] );
/* ніжного правого кута */
gotoxy(xul,ylr); putch( boxcar[btype - 1][4] );

    }
  }
//---------------------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
int c,ch;
   int btype ;
   int xul,yul;      /* кординати лівого кута рамки */
   int xlr,ylr;      /* кординати правого кута рамки */
   int N,n;   /* N -номер вікна, n –кількість символів у вікні */
   char S_1[7],S_2[15],S_3[15],S_4[8]; /* ліворуч Ctr+shift -> шріфт EN*/
                                       /* праворуч Ctrl+Shift -> шріфт Ru */
p1: clrscr();  btype = 2;
 gotoxy(3,22); textcolor(RED); cprintf("ESC");
 gotoxy(7,22); textcolor(GREEN); CharToOem("Выход\n",S_1); cprintf(S_1);
 gotoxy(1,23); textbackground(LIGHTCYAN); textcolor(RED);
      CharToOem("Лабораторна    \n",S_2); cprintf(S_2);
    gotoxy(1,24); CharToOem("работа  № 8 ->1 \n",S_3); cprintf(S_3);
  textmode(LASTMODE); gotoxy(1,1);
      N = 1; xul =1; yul = 1;  ch = (dx - 2) * (dy -1) + 2 ;
   do
     { xlr = xul + dx;  n = 0;
       ylr = yul + dy;
   textcolor(CYAN);
```

```
        my_box(xul,yul,xlr,ylr,btype);
        window(xul + 1,yul + 1,xlr - 1,ylr - 1);
         gotoxy(0,0);
        do
        { c = getch();
            if( c != 0x1b )
             {   textbackground(BLUE); textcolor(YELLOW);
                 putch(c);
                  n++;
             }
              else
                goto p2;
        } while( n <= ch );
      window(1,1,80,25);
          gotoxy(xlr - 17,ylr + 1);  textbackground(LIGHTGRAY);
          textcolor(MAGENTA); cprintf(" ASCII "); CharToOem("код
\n",S_4);cprintf(S_4);
          gotoxy(xlr - 17,ylr + 2);  textbackground(GREEN);
    textcolor(RED); cprintf("  "); putch(c);
                              cprintf("= 0x%x  ",c);
         textmode(LASTMODE);
          N++;   btype --;
        xul = xlr + 1;     /* кординати для window(1,1,80,25) */
          yul = ylr + 1;
       } while( N <= 3);
      window(1,1,80,25);
      goto p1;
    p2:          ;
         return 0;
    }  /*end main*/
    //---------------------------------------------------------------------------
```

Методика виконання задачі Б.

Порядок дій і команд при виконанні програмування задачі Б

Крок 1. За допомогою кнопку ***New*** відкрийте вікно ***New Items*** і виберіть ***Console Wizard*** і для опції ***Source Type*** задайте режим ***C***++. У результаті ***C***++ ***Builder*** відкриє шаблон тексту вихідної прикладної програми в такому вигляді:

```
//---------------------------------------------------------------------------
#pragma hdrstop
//---------------------------------------------------------------------------
#pragma argsused
```

```
int main(int argc, char* argv[])
{
      return 0;
}
//---------------------------------------------------------------------------
```

Крок 2. Додайте у відповідні місця шаблона програми вказівки препроцессора, оголошення й оператори у відповідності з наступним текстом програми

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
/*=============================================*
 *      Креслення рамки за допомогою           *
 *    ASCII кодів псевдографічних символів     *
 *=============================================*/
#include <system.hpp>
#include <stdio.h>
#include <conio.h>
 void draw_border( int startx,int starty,int endx,int endy)
 { register  int i;
  textcolor( LIGHTCYAN);
  gotoxy(startx,starty); cprintf("%c",218); /* putchar( 218 );*/
  gotoxy(startx,endy);   cprintf("%c",192); /* putchar( 192 );*/
  gotoxy(endx,starty);   cprintf("%c",191); /* putchar( 191 );*/
  gotoxy(endx,endy);     cprintf("%c",217); /* putchar( 217 );*/
   getch();
  for( i = startx + 1; i < endx ; i++ )
   {   textcolor( LIGHTRED);
     gotoxy(i,starty); cprintf("%c",196); /*putchar( 196 );*/
     gotoxy(i,endy);   cprintf("%c",196);  /*putchar( 196 );*/
     getch();
   }
      for( i = starty + 1; i < endy ; i++ )
       {   textcolor( LIGHTMAGENTA);
         gotoxy(startx,i); cprintf("%c",179); /* putchar( 179 );*/
         gotoxy(endx,i);   cprintf("%c",179); /* putchar( 179 );*/
        getch();
       }
   textmode(LASTMODE);

 }
//---------------------------------------------------------------------------
#pragma argsused
```

```
int main(int argc, char* argv[])
{
 int lux,luy;   /* Лівого верхнього кута рамки */
 int rdx,rdy;   /* Правого ніжнього кута рамки */
 char S_1[15],S_2[15],S_3[15],S_4[20],S_5[22],S_6[22],S_7[22];
 char S_8[30],S_9[30],S_10[30],S_11[30];
                              /* ліворуч Ctr+shift -> шріфт EN*/
                              /* праворуч Ctrl+Shift -> шріфт Ru */
m1 :  clrscr();
     gotoxy(1,23); textbackground(LIGHTCYAN); textcolor(RED);
     CharToOem("Лабораторная    \n",S_1); cprintf(S_1);
   gotoxy(1,24); CharToOem("работа № 8->2  \n",S_2); cprintf(S_2);
   gotoxy(30,1);  textbackground(GREEN); textcolor(RED);
   CharToOem("ВВЕДІТЬ\n",S_3); cprintf(S_3);
   gotoxy(1,3);   textbackground(LIGHTGRAY); textcolor(LIGHTBLUE);
   CharToOem("Кординати лівого\n",S_4); cprintf(S_4);
   gotoxy(1,4);
   CharToOem("верхнього кута рамки \n",S_5); cprintf(S_5);
 gotoxy(28,4);   textmode(LASTMODE);  textcolor(LIGHTGREEN);
            cprintf(" X="); scanf("%d", &lux);
 gotoxy(37,4);     cprintf(" Y="); scanf("%d", &luy);
   gotoxy(40,23);  textbackground(LIGHTGRAY); textcolor(LIGHTBLUE);
   CharToOem("Кординати правого\n",S_6); cprintf(S_6);
    gotoxy(40,24);
    CharToOem("ніжнього кута рамки \n",S_7); cprintf(S_7);
 gotoxy(62,24);  textmode(LASTMODE);   textcolor(LIGHTGREEN);
             cprintf(" X="); scanf("%d", &rdx);
 gotoxy(72,24);     cprintf(" Y="); scanf("%d", &rdy);
  if( rdx < lux)
  {  textcolor(LIGHTBLUE);
   gotoxy(40,10);
   CharToOem("Ви помилились при введенні кординати X \n",S_8); cprintf(S_8);
   gotoxy(40,11);
   CharToOem("ніжнього правого кута рамки \n",S_9); cprintf(S_9);
   getch();  textmode(LASTMODE);
     goto m1;
  }
      if( rdy < luy )
      {   textcolor(LIGHTBLUE);
       gotoxy(40,12);
    CharToOem("Ви помилились при введенні кординати Y \n",S_10);
cprintf(S_10);
   gotoxy(40,13);
   CharToOem("ніжнього правого кута рамки \n",S_11); cprintf(S_11);
        cprintf(" ");
       getch();   textmode(LASTMODE);
        goto m1;
      }
```

```
        lux =( lux > 70 ) ? 70 : lux;
        luy =( luy > 20 ) ? 18 : luy;
        rdx =( rdx > 80 ) ? 80 : rdx;
        rdy =( rdy > 20 ) ? 20 : rdy;
      draw_border( lux,luy,rdx,rdy );
          return 0;
    } /* end main() */
//---------------------------------------------------------------------------
```

Виконання прикладної програми в MS DOS до задачи Б з лабораторної роботи забезпечується вікном, яке зображено на рис. 2.1.

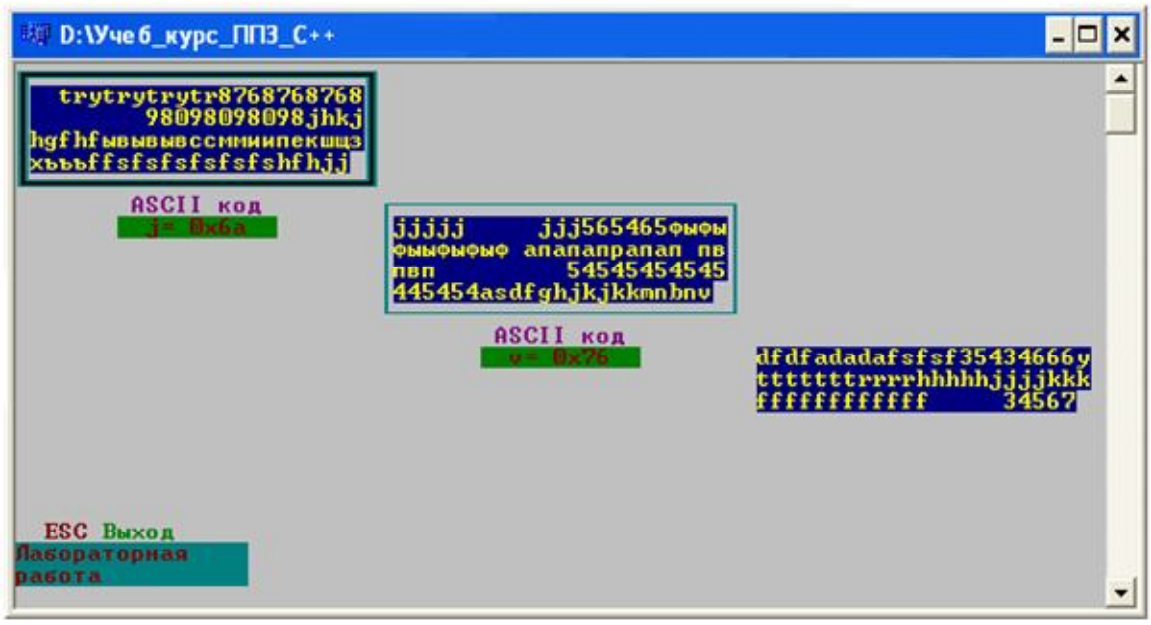

Рис. 2.1. Зображення вікна С++ програми до задачи Б.

Контрольні запитання до лабораторної роботи № 2

1. Пояснить блок-схему виконання прикладної програми до задачі А.
2. Пояснить, яка функція показує ***ScanAsCII Code*** після натискання клавіши.
3. Як налаштовується ***C++ Builder*** для одержання коду консольної прикладної ***C++*** програми для виконання у ***MS DOS***.
4. Поясніть алгоритм рисування рамок за допомогою псевдографічних символів.
5. Яка функція забезпечує у вікнах програми написи підказок кирилицею.

Лабораторна робота № 3

Техніка використання технології Drag & Doc і плаваючих вікон в прикладних C++ програмах

Ціль виконання лабораторної роботи. Передбачається виконання таких навчальних задач:

- вивчення правил створення прикладної програми з декількома відкритми документами;
- вивчення техніки вбудовання віконних об'єктів на основі технології ***Drag & Doc.***

Особливості технології *Drag & Doc*

Технологія ***Drag & Doc*** забеспечує плаваючі вікна в прикладній ***C***++ програмі. Це забезпечується за рахунок переміщення і вбудовання вікон в інші вікна прикладної програми. Кожне вбудоване вікно може мати вкладку з відповідною назвою, що забезпечує більше вільного місця на екрані і добрий інтерфейс для користувача.

Постановка задачі по програмуванню у лабораторній роботі

Необхідно для ***Windows*** створити прикладну ***C***++ програму з плаваючими вікнами ***Fdoc***, яки можна вбудувати у одне вікно з назвою ***Fmain*** (рис. 3.1).

Завдання для аналізу і обробки результатів роботи запрограмованої C++ програми

Після запуску в роботу ***C***++ програми лабораторної роботи № 3 для захисту результатів програмування необхідно виконати таке:

- нарисувати блок-схему алгоритму з роботи програми лабораторної роботи № 3;
- нарисувати блок-схему алгоритму по налаштуванню властивостей до вікон ***Fdoc*** та ***Fmain***.

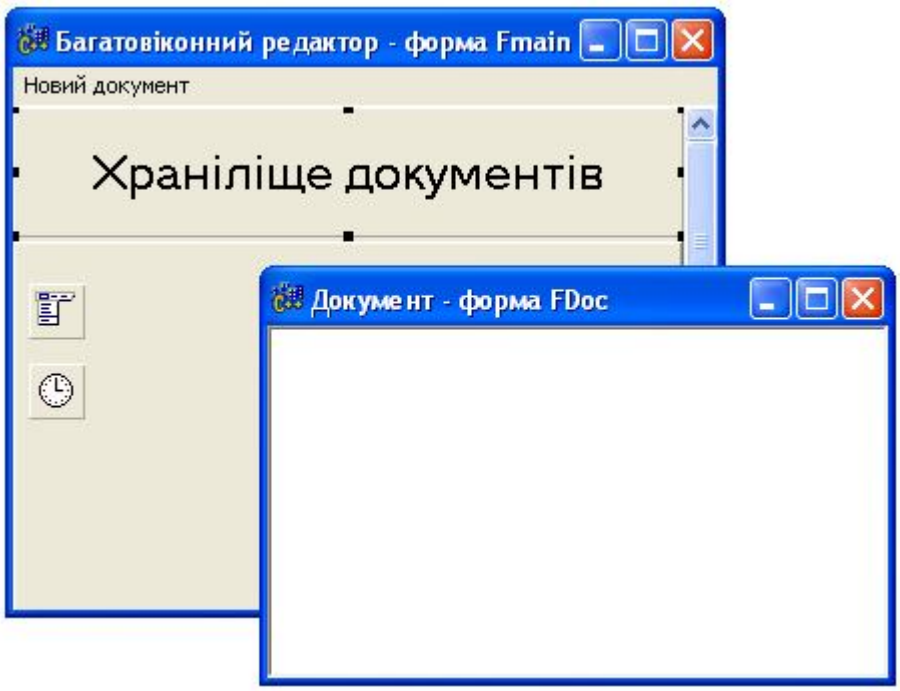

Рис. 3.1.

⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇

Методика виконання задачі.

Порядок дій і команд при виконанні програмування задачі

Крок 1. Для форми ***Fmain*** файли листингов коду програми мають такі назви:

Umain.cpp, ***Umain.h***.

Крок 2. Для форми документу ***FDoc*** файли листингов коду програми мають наступні назви: ***Udoc.cpp, Udoc.h, Pdoc.cpp.***

Крок 3. Файли модулів реалізації і заголовних файлів мають наступні коди:

Pdoc.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USEFORM("Umain.cpp", Fmain);
USEFORM("Udoc.cpp", FDoc);
//---------------------------------------------------------------------------
```

```
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TFmain), &Fmain);
                 Application->CreateForm(__classid(TFDoc), &FDoc);
                 Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        return 0;
}
//---------------------------------------------------------------------------
```

Umain.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Umain.h"
#include "Udoc.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFmain *Fmain;
TList * LDoc;
//---------------------------------------------------------------------------
__fastcall TFmain::TFmain(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TFmain::MNewClick(TObject *Sender)
{
 TFDoc * New = new TFDoc(this);
 LDoc->Add(New);
 New->Caption = "Документ "+IntToStr(LDoc->Count);
}
//---------------------------------------------------------------------------
void __fastcall TFmain::FormCreate(TObject *Sender)
{
 LDoc = new TList();
}
//---------------------------------------------------------------------------
```

Umain.h

```
//---------------------------------------------------------------------------
#ifndef UmainH
```

```
#define UmainH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
//---------------------------------------------------------------------------
class TFmain : public TForm
{
__published:        // IDE-managed Components
        TPageControl *PageControl1;
        TPanel *Panel1;
        TMainMenu *MainMenu1;
        TMenuItem *MNew;
        void __fastcall MNewClick(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
private:     // User declarations
public:                 // User declarations
        __fastcall TFmain(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TFmain *Fmain;
//---------------------------------------------------------------------------
#endif
```

Udoc.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Udoc.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFDoc *FDoc;
//---------------------------------------------------------------------------
__fastcall TFDoc::TFDoc(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

Udoc.h

```
//---------------------------------------------------------------------------
#ifndef UdocH
#define UdocH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
```

```
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TFDoc : public TForm
{
__published:        // IDE-managed Components
        TMemo *Memo1;
private:     // User declarations
public:              // User declarations
        __fastcall TFDoc(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TFDoc *FDoc;
//---------------------------------------------------------------------------
#endif
```

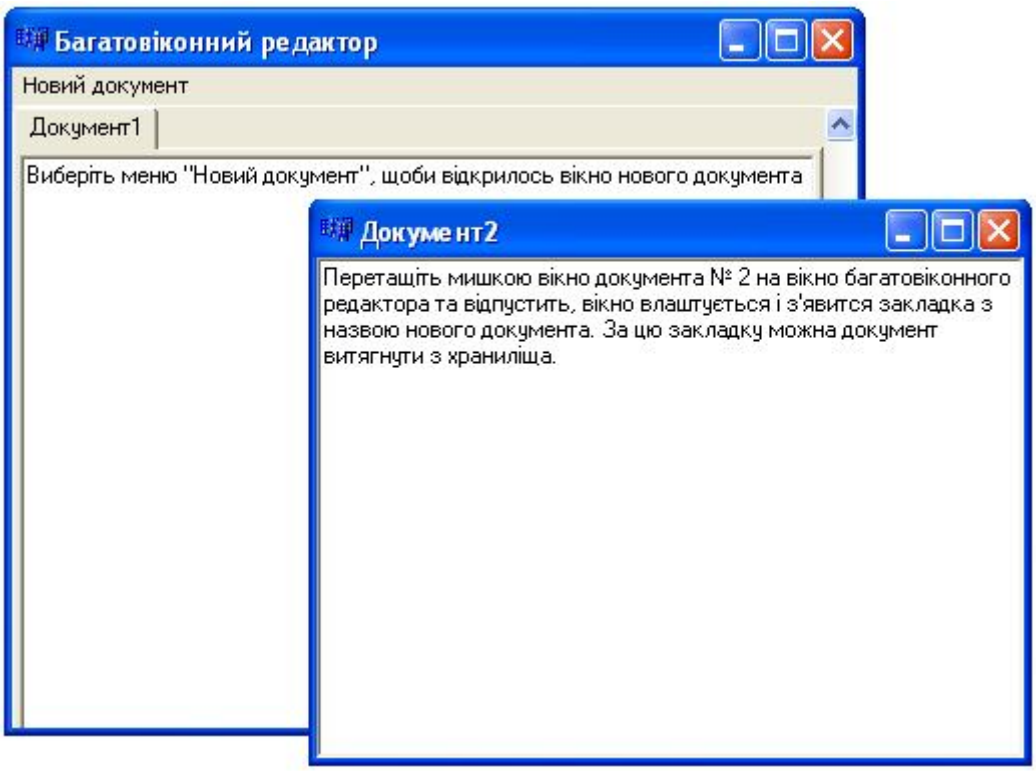

Рис. 3.2. Результати роботи прикладної програми лабораторної роботи № 3.

Контрольні запитання до лабораторної роботи № 3

1. Поясніть виконання алгоритму до **Pdoc.cpp.**
2. Поясніть виконання алгоритму до **Umain.cpp.**
3. Поясніть виконання алгоритму до **Umain.h.**
4. Поясніть виконання алгоритму до **Udoc.cpp.**
5. Поясніть виконання алгоритму до **Udoc.h.**

Лабораторна робота № 4

Алгоритми з розробки і використання фреймів у C++ Builder при створенні прикладної програми

Ціль виконання лабораторної роботи. Передбачається виконання таких навчальних задач:

- вивчення правил і методики розробки у ***C++ Builder*** відповідного фрейму для подальшого використання в прикладних C++ програмах;

- отримання навичок практичної роботи з командами для сформованого фрейму;

- практичне вивчення правил і команд по заповненню текстів операторів в шаблони програмних модулів до розробленого фрейму;

- практична робота з набором файлів до фрейму і прикладної програми для виконання умовної компіляції;

- освоєння техніки застосування розробленого фрейму при створенні прикладної C++ програми.

- вивчення команд і методики розміщення розробленого фрейму в Депозитарій інтегрованого програмувального середовища C++ Builder;

- вивчення властивостей фрейму та форми вікна програми в залежності від налаштування у Депозитарії радіо-кнопок до режиму використання фрейму.

Загальні зведення до розробки і використання фреймів у C++ Builder

В C++ Builder у бібліотеці VCL є компонента ***Frame,*** котра дозволяє швидко створювати в одному стилі необхідні елементи у вікнах прикладної програми. В інтегрованому програмувальному середовищі ***Frame*** – це поле без розміточної сітки, але яке має своє вікно і займає середнє положення між панелями і формою прикладної C++ програми. Фрейм має з формою в C++Builder таки однакові властивості:

- ✓ фрейм спочатку проектується окремо і має своє вікно стандартного формату;

- ✓ кожний створений фрейм в C++ Builder має свій відповідний програмний модуль, тобто файл з розширенням **.cpp ;**
- ✓ фрейми мають можливість наслідувати властивості ширше і більше ніж у формах при створенні прикладної програми;
- ✓ розроблені фрейми для використання можна розміщувати у Депозитарій інтегрованого програмувального середовища і при створенні прикладної програми використовувати їх подібно до шаблонів з різних віконних форм;
- ✓ фрейм відрізняється збереженням наслідування властивостей більше, ніж форма вікна програми.

Фрейм зі своїм вікном після створення веде себе, як деяка панель і не може бути самостійним вікном ***Windows*** тому що, обов'язково фрейм необхідно встановлювати на відповідну форму вікна програми або на контейнер. Також фрейми мають свої властивості, методи та події, яки подібні панелям(компонентам). Таким чином розроблений фрейм треба розглядати на формі вікна як деяку панель до фрагменту прикладної програми, яку можна переносити на інші форми та програми зі збереженням (наслідуванням) властивостей.

Починається розробка (проектування) нового фрейму командами:
перший варіант – ***File/New/Frame*** ;
другій варіант – ***File/New/Other*** та потім у вікні Депозитарію на сторінці ***New*** потрібно обрати значок ***Frame***. В цих обох випадках відкриється вікно фрейму без розміточної сітки, подібне вікну форми яка має розміточну сітку і також у вікні "Редактор Коду " одночасно з'явиться для фрейму шаблон програмного коду з такою структурою:

```
…
#include "Unit2.h"
….
TFrame2 *Frame2;

/* В дане місце розміщуються об'яви до типів даних констант та об'яви до типів
даних змінних, до яких не повинно бути доступу з інших програмних модулів,
тому що вони мають бути одними до усіх об'єктів фрейму. Також в цьому місці
```

повинні вказуватися функції, яки далі запишуться у файл ***Unit2.h*** та інші необхідні об'яви функцій */

```
// Шаблон конструктора
_fastcall TFrame2::TFrame2(TComponent *Owner)
: TFrame(Owner)
{
}
```

Одночасно у вікні "Редактор Коду " також створюється і шаблон для файлу ***Unit2.h*** у такому вигляді:

```
….
// Об'яви до класу фрейму
Class TFrame2 : public TFrame
{
published: //IDE-manager Components

/* В дане місце C++ Builder записуються об'яви до компонент, яки розміщуються
на полі вікна фрейму. Додавати інші записи вручну забороняється. */

private:  //Users declarations

/* Зачинений розділ класу. Сюди можна записувати об'яви до змінних і функцій,
яки включаються до класу фрейму, але недосяжних для інших модулів функцій. */

public:      //Users declarations

/* Відкритий розділ класу. Сюди можна записувати об'яви до змінних і функцій,
яки включаються до класу фрейму, але досяжних для інших модулів функцій. */

// Об'ява конструктора
_fastcall TFrame2( TComponent *Owner) ;
};
…..
```

При відкритті фрейму можна з бібліотеки компонент VCL переносити обрану компоненту на поле вікна, подібно як це робиться до поля вікна форми прикладної C++ програми. Майже часто у фреймах встановлюються компоненти, яки наведені у таблиці 4-1.

Після створення відповідного фрейму його можна розташувати в Депозитарій або у палітру бібліотеки VCL, щоби дали використовувати в різних прикладних C++ програмах. Для цього потрібно на фреймі виконати команду правою клавішею мишки і з контекстного меню команд обрати

Add To Repositary або ***Add To Palette***. Якщо розмісти створений фрейм в Депозитарій, тоді при внесенні змін у фрейм за рахунок властивостей наслідування також будуть виникати зміни і в усіх прикладних програмах, створених з участю даного фрейму. Таким чином можна створювати серію прикладних програм з одним стилем виконання форми вікна С++ програми.

Таблиця № 4-1.

Назва компоненти	**Страніца з VCL**	**Призначення компоненти**
GroupBox	Standart	Візуальна компонента – контейнер для об'єднання набору елементів, розташованих на полі контейнера.
Edit	Standart	Візуальна компонента для введення користувачем одної строки тексту або для відображення тексту.
Button	Standart	Візуальна компонента для створення кнопок, котрими користувач виконуватиме команди.
OpenPictureDialog	Dialogs	Невізуальна компонента для створення вікна діалогу «Відкрити рисунок».
Application-Events	Additional	Невізуальна компонента для перехвату подій в прикладній програмі.
Image	Additional	Візуальна компонента для відображення графічних зображень: піктограмки, значки, бітові матриці і метадані.
Label	Standart	Візуальна компонента для розміщення на формах та інших контейнерах написів, котрі користувач при роботі програми змінювати не буде.
Frame	Standart	Візуальна компонента – панель з вікном та можливостями наслідування властивостей компонент, розташованих на полі панелі.

В ***C***++ ***Builder*** вхід в Депозитарій виконується командою ***File / New / Other*** і в результаті відривається діалогове вікно ***New Items***, де можна обрати готові шаблони форм до ***C***++ програм чи програмні конструктори для виконання розробок, або можна зберігати готові фрейми чи форми. Дуже часто при створенні форми або фрейму з багатьма компонентами витрачається багато часу. При повторних використаннях таких форм і фреймів потрібно використовувати Депозитарій, щоби не витрачати додаткові години на розробку нової прикладної ***C***++ програми.

В діалоговому вікні ***New Items*** знизу розташовані таки радіо-кнопки:

- ***Copy*** (копіювати) при використанні даного режиму фрейм і файли копіюються до форми і далі при змінах на формі або на фреймі ніякого зв'язку не виникає;

- ***Inherit*** (наслідування) при використанні даного режиму для фрейму в проекті буде отримана форма вікна з наслідуванням властивостей фрейму, що означає наступне:

  - ✓ при внесенні змін у фрейм, який збережено у Депозитарій, буде викликати відповідні зміни у фреймі на формі після перекомпіляції проекту файлів;
  - ✓ при внесенні змін у фрейм на формі, який збережено у Депозитарій, не буде викликати відповідні зміни у самому фреймі після перекомпіляції проекту файлів;

- ***Us*** (використання) в даному режимі в проекті програми буде отримана форма вікна разом з залежним наслідуванням властивостей фрейму, що означає наступне:

  - ✓ при внесенні змін у фрейм на формі вікна проекту програми одночасно з'являться зміни після виконання перекомпіляції і у фрейму, який знаходиться у Депозитарії.

⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇

Постановка задачі по програмуванню у лабораторній роботі

У лабораторній роботі № 4 необхідно виконати наступні дві задачі:

Задача А. Для використання в прикладній C++ програмі спочатку необхідно розробити фрейм відповідно до рис. 4-1 і зберегти файли його програмних модулів в папку ***LAB*4,** де створюємо папку ***LAB*4_A** і в неї зберігаємо усі файли проекту ***P_Lab*4_A**. Після створення фрейму відповідно до рис. 4-1 необхідно за допомогою даного фрейму розробити вікно форми до прикладної ***C***++ програми (рис. 4-2), де при виконанні команд в діалогових вікнах повинна бути можливість для вибору файлу з рисунком і

отриманням результату роботи ***C***++ програми у вигляді рис. 4-3 та рис. 4-4. Усі вихідні файли прикладної ***C***++ програми і файли проекту ***P_Lab*4_A** зберігаємо в папці ***LAB*4_A**.

Задача Б. Необхідно фрейм, який розроблено у задачі А, розташувати у Депозитарій ***C***++ ***Builder*** і з його допомогою створити нову додаткову прикладну ***C***++ програму та за допомогою якої перевірити режими наслідування властивостей при внесенні змін у фрейму, який буде розташованим у Депозитарій. Усі вихідні файли нової ***C***++ програми і файли проекту ***P_Lab*4_B** зберігати в папку ***LAB*4_B** з папці ***LAB*4** .

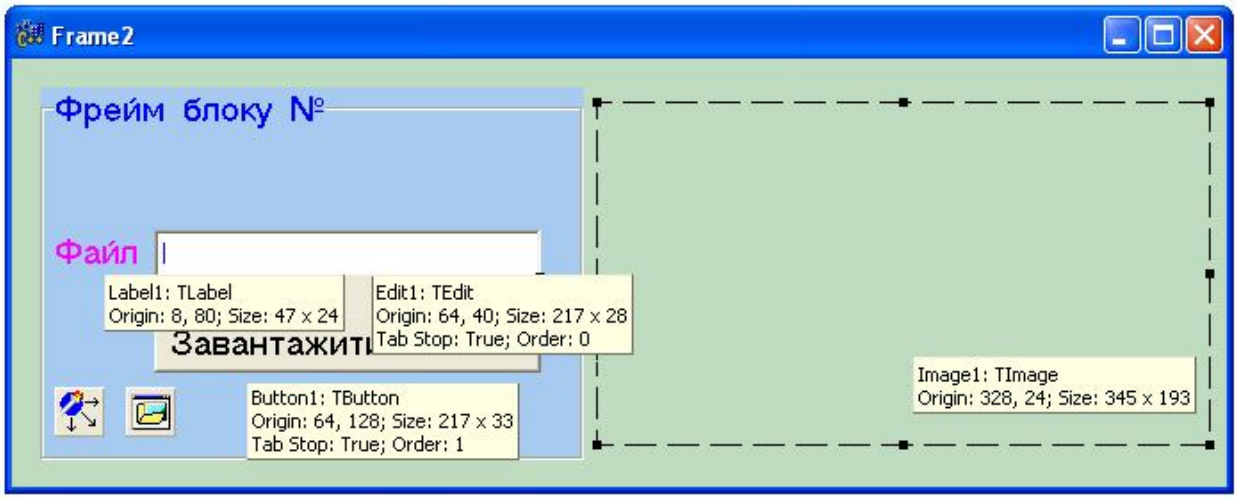

Рис. 4-1. Структура елементів для розроблення фрейму.

Завдання для аналізу і обробки результатів роботи запрограмованої C++ програми

Після запуску в роботу ***C***++ програм до задач А та Б з лабораторної роботи № 4 для захисту результатів програмування необхідно виконати таке:

- нарисувати блок-схему алгоритму до виконання команд розробленого фрейму з проекту ***P_Lab*4_A** лабораторної роботи № 4;
- нарисувати структурну схему дій і команд для використання фрейму з вікна Депозитарію при створенні прикладної С++ програми до проекту ***P_Lab*4_B** лабораторної роботи № 4.

- створити скрин-шоти по результатах досліджень з використання фрейму з Депозитарію.

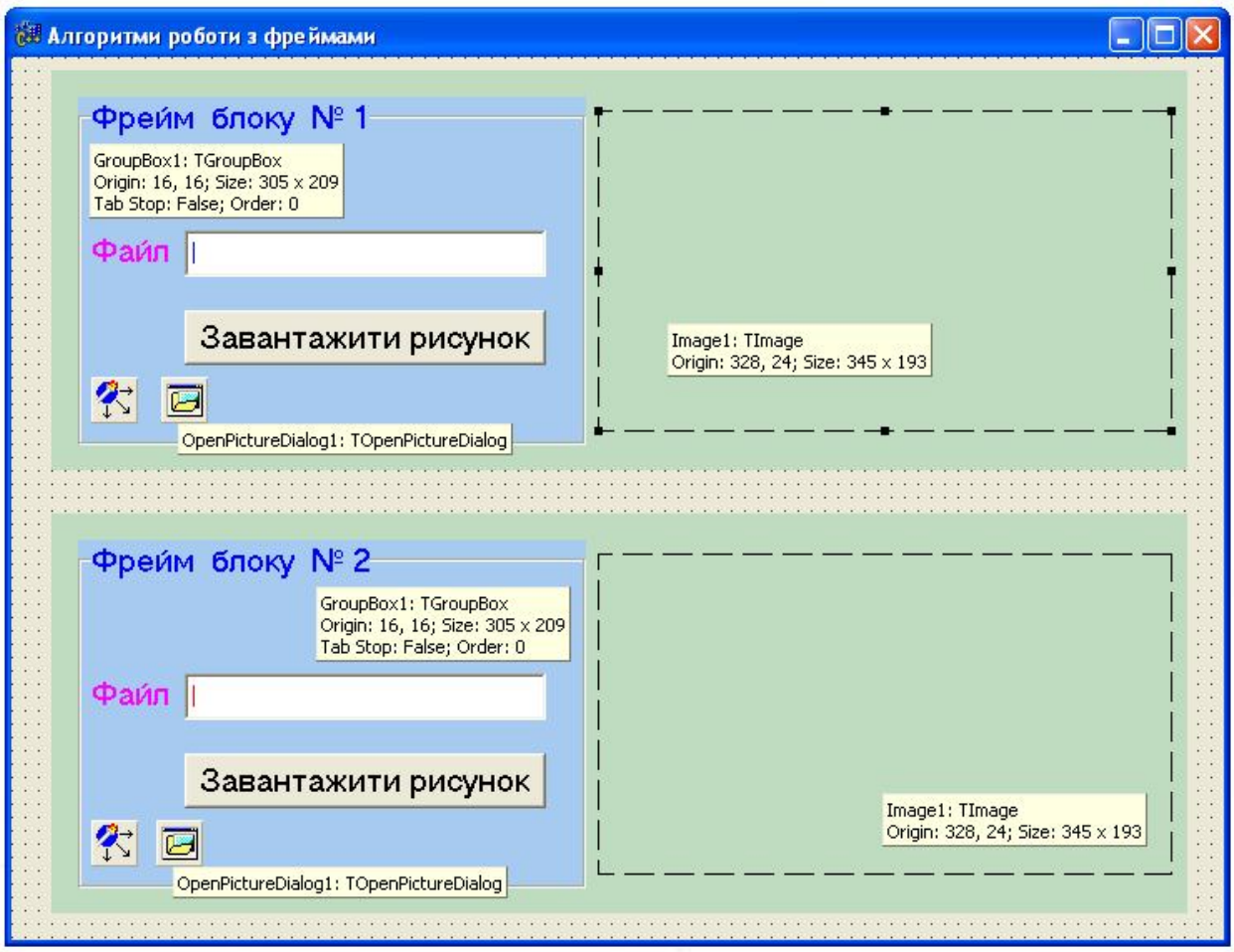

Рис. 4-2. Структура елементів на формі вікна прикладної С++ програми.

Методика виконання задачі А.

***Порядок дій і команд при виконанні програмування задачі А з розробкою фрейму завданої структури*:**

Крок 1. Спочатку необхідно активізувати вікно фрейму і для цього виконуємо наступне:

➢ Виконайте команду ***File / New / Frame***, щоб з'явилася нове вікно з назвою ***Frame*2** (рис. 4-5).

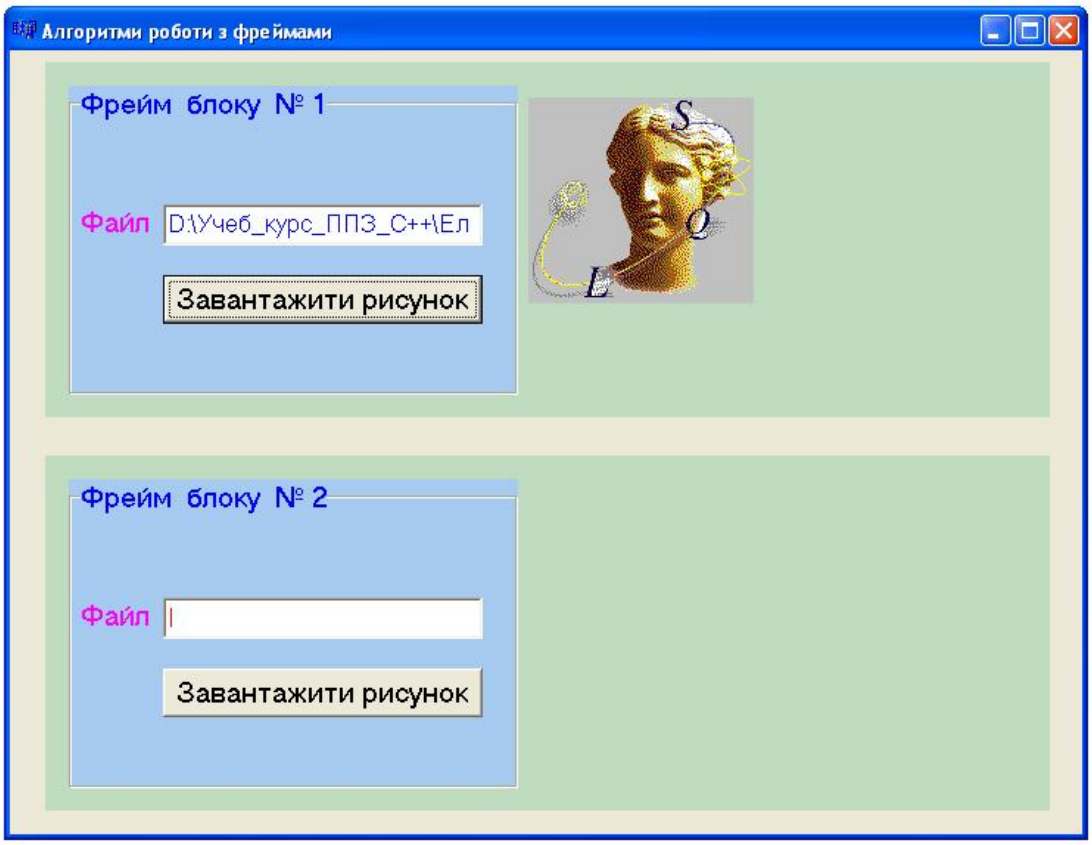

Рис. 4-3. Результат виконання команд у фреймі блоку № 1 C++ програми.

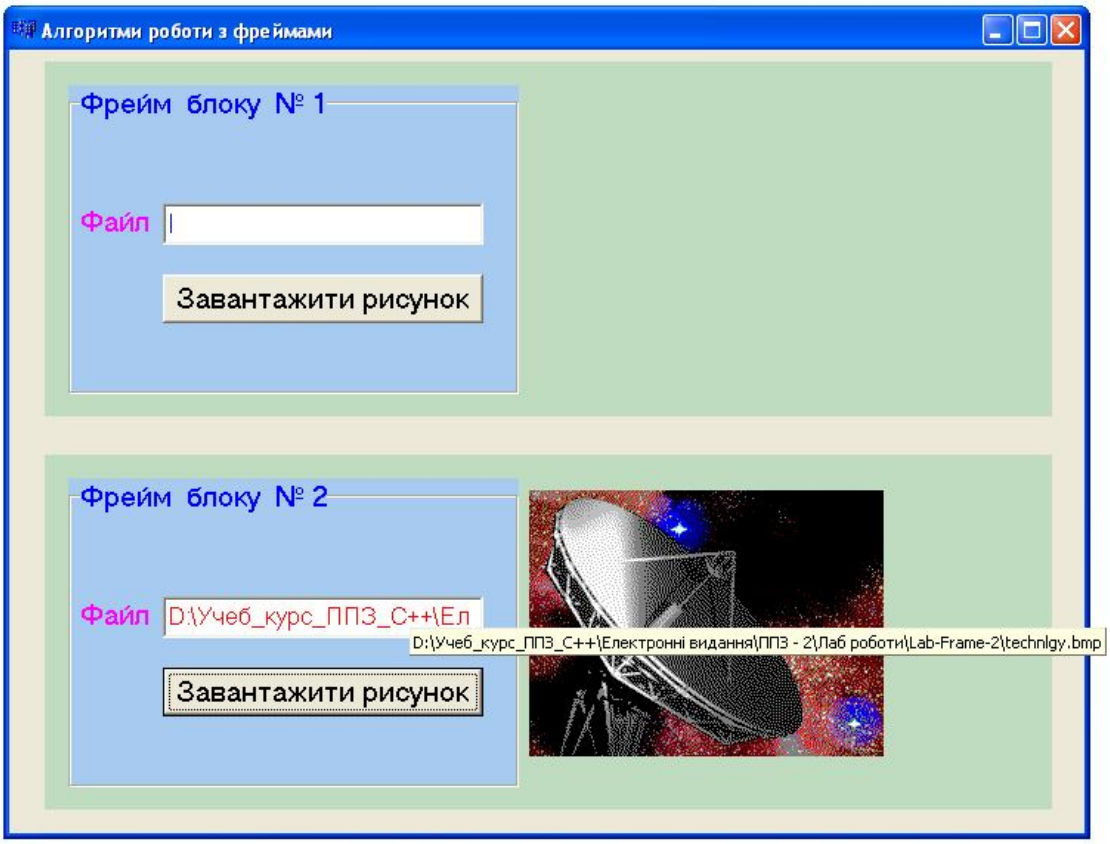

Рис. 4-4. Результат виконання команд у фреймі блоку № 2 C++ програми.

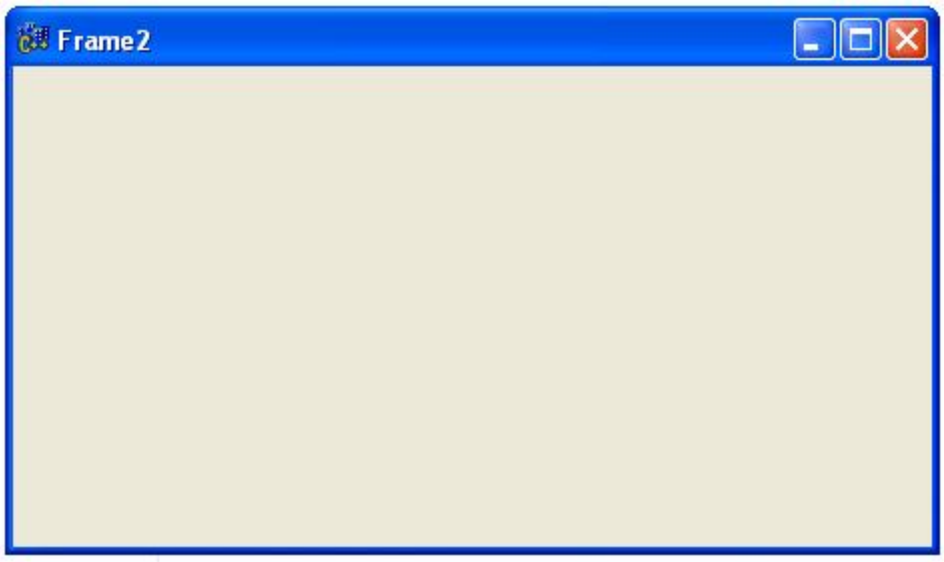

Рис. 4-5. Зображення вікна нового фрейму.

Крок 2. Починаємо наповнювати поле нового фрейму ***Frame*2** компонентами з бібліотеки ***VCL*** і для цього виконуємо наступне :

- На сторінці ***Standart*** в ***VCL*** мишкою визначаємо компоненту ***GroupBox*1** і встановлюємо на поле фрейму;
- На сторінці ***Standart*** в ***VCL*** мишкою визначаємо компоненту ***Edit*1** і встановлюємо на поле фрейму;
- На сторінці ***Standart*** в ***VCL*** мишкою визначаємо компоненту ***Button*1** і встановлюємо на поле фрейму;
- На сторінці ***Standart*** в ***VCL*** мишкою визначаємо компоненту ***Label*1** і встановлюємо на поле фрейму;
- На сторінці ***Dialogs*** в ***VCL*** мишкою визначаємо невізуальну компоненту ***OpenPictureDialog*** і встановлюємо на поле фрейму;
- На сторінці ***Additional*** в ***VCL*** мишкою визначаємо невізуальну компоненту ***Application-Events*** і встановлюємо на поле фрейму;
- На сторінці ***Additional*** в ***VCL*** мишкою визначаємо візуальну компоненту ***Image1*** і встановлюємо на поле фрейму.

Після виконання, вище вказаних дій, на полі фрейму компоненти повинні розміщуватися відповідно до рис. 4-6.

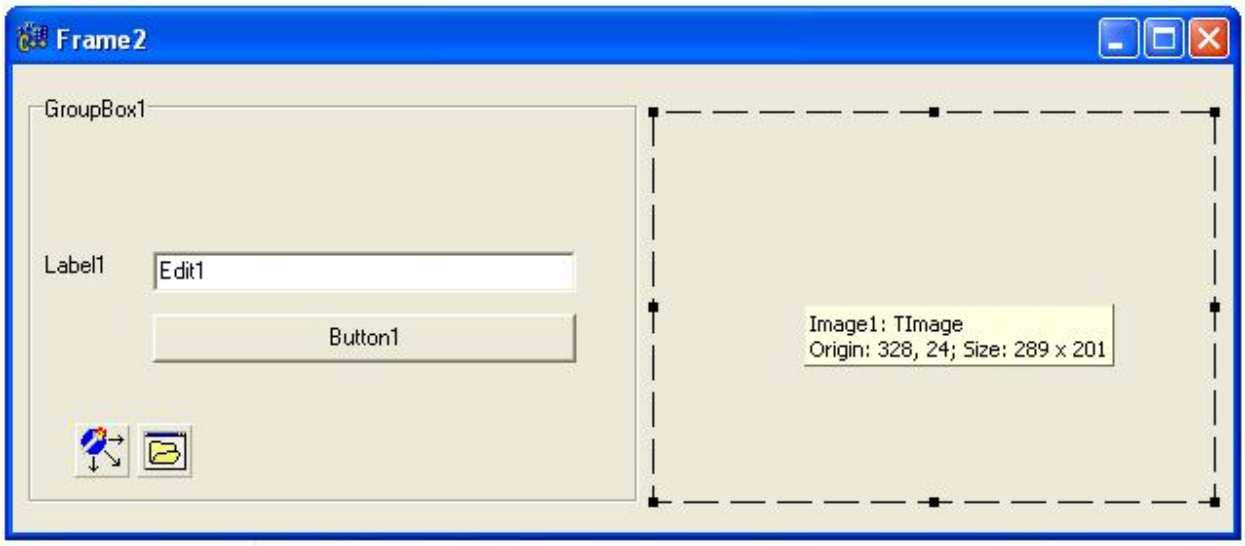

Рис. 4-6. Зображення вікна фрейму після розміщення компонент.

Крок 3. Далі необхідно у вікні інспектора об'єктів виконати наступні налаштування для компонент, розміщених на полі фрейму:

- На компоненту ***OpenPictureDialog*** встановлюємо мишкою маркерну рамку і переходимо у вікно інспектора об'єктів, де виконуємо ***Properties*** / ***Filter*** і біля напису ***All files****.* робимо щиглика мишкою на віконці з трьома крапками для відкриття вікна ***FilterEditor*** , де замість назви « ***All files*** » записуємо назву «Всі файли».
- На компоненту ***Edit*1** встановлюємо мишкою маркерну рамку і переходимо у вікно інспектора об'єктів, де виконуємо ***Properties*** / ***ShowHint*** і замінюємо напис ***false*** зі списку на ***true*** щоби показувався ярличок підказки. Для цього у полі ***Hint*** задайте назву «Вибір файлу»;
- На компоненту ***Button*1** встановлюємо мишкою маркерну рамку і переходимо у вікно інспектора об'єктів, де виконуємо ***Properties*** / ***ShowHint*** і замінюємо напис ***false*** зі списку на ***true*** щоби показувався ярличок підказки. Для цього у полі ***Hint*** задайте назву «Вибір файлу з каталогу» ;
- Для відображення підказок з ***Hint*** необхідно виконати налаштування обробника події. Для цього зліва зверху у вікні ***Object TreeView*** робимо щиглика мишкою на назві ***ApplicationEvents***1 для маркування та далі на закладці ***Events*** обираємо подвійним щигликом мишки обробник події ***OnShowHint*** для відкриття у вікні «Редактор Коду» шаблону функції з

обробки події. Цей шаблон необхідно доповнити операторами і він повинен бути таким:

```
//---------------------------------------------------------------------------
void __fastcall TFrame2::ApplicationEvents1ShowHint(AnsiString &HintStr,
    bool &CanShow, THintInfo &HintInfo)
{
if(HintInfo.HintControl = = Edit1)
      if(Label1->Canvas->TextWidth(Edit1->Text) > Edit1->Width)
      {HintStr = Edit1->Text;
      ApplicationEvents1->CancelDispatch();
      }
}
//---------------------------------------------------------------------------
```

Крок 4. Далі необхідно у програмному модулі ***Unit2_frame.cpp*** зробити об'яву глобальної змінної з назвою ***FileName*** і такий запис повинен мати такий вигляд:

```
 //---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit2_frame.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFrame2 *Frame2;
static AnsiString FileName;
//---------------------------------------------------------------------------
```

Крок 5. Далі необхідно виконати налагодження функції з обробки події до кнопки ***Button*1** і для цього виконуємо таки дії:

- Для цього зліва зверху у вікні ***Object TreeView*** робимо щиглика мишкою на назві ***Button*1** для маркування та далі на закладці ***Events*** обираємо подвійним щигликом мишки обробник події ***OnClick*** для відкриття у вікні «Редактор Коду» шаблону функції з обробки події.

Цей шаблон необхідно доповнити операторами і він повинен бути таким:

```
//---------------------------------------------------------------------------
void __fastcall TFrame2::Button1Click(TObject *Sender)
{
if(OpenPictureDialog1->Execute() )
{ Edit1->Text = OpenPictureDialog1->FileName;
FileName = OpenPictureDialog1->FileName;
Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
}
}
//---------------------------------------------------------------------------
```

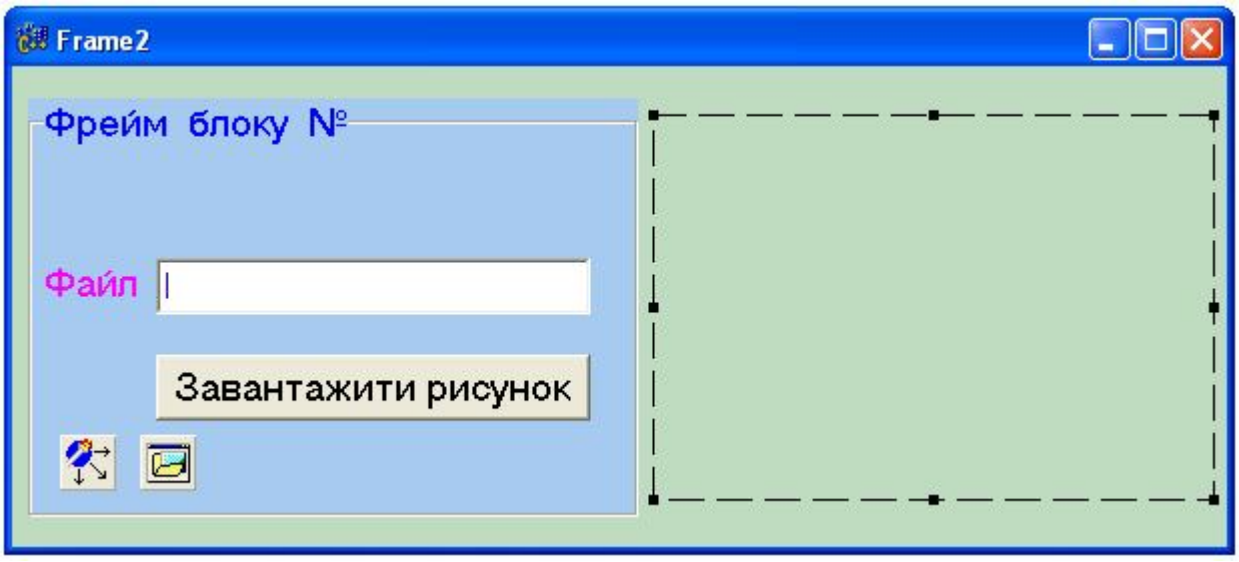

Рис. 4-7. Зображення вікна фрейму після визначення кольорів.

Крок 6. Далі необхідно виконати налагодження обробки події до компоненти ***Edit*1** і для цього виконуємо таки дії:

- Для цього зліва зверху у вікні ***Object TreeView*** робимо щиглика мишкою на назві ***Edit*1** для маркування та далі на закладці ***Events*** обираємо подвійним щигликом мишки обробник події ***OnExit*** для відкриття у вікні «Редактор Коду» шаблону функції з обробки події. Цей шаблон необхідно доповнити оператором і він повинен бути таким:

```
//---------------------------------------------------------------------------
void __fastcall TFrame2::Edit1Exit(TObject *Sender)
{
FileName = Edit1->Text;
}
//---------------------------------------------------------------------------
```

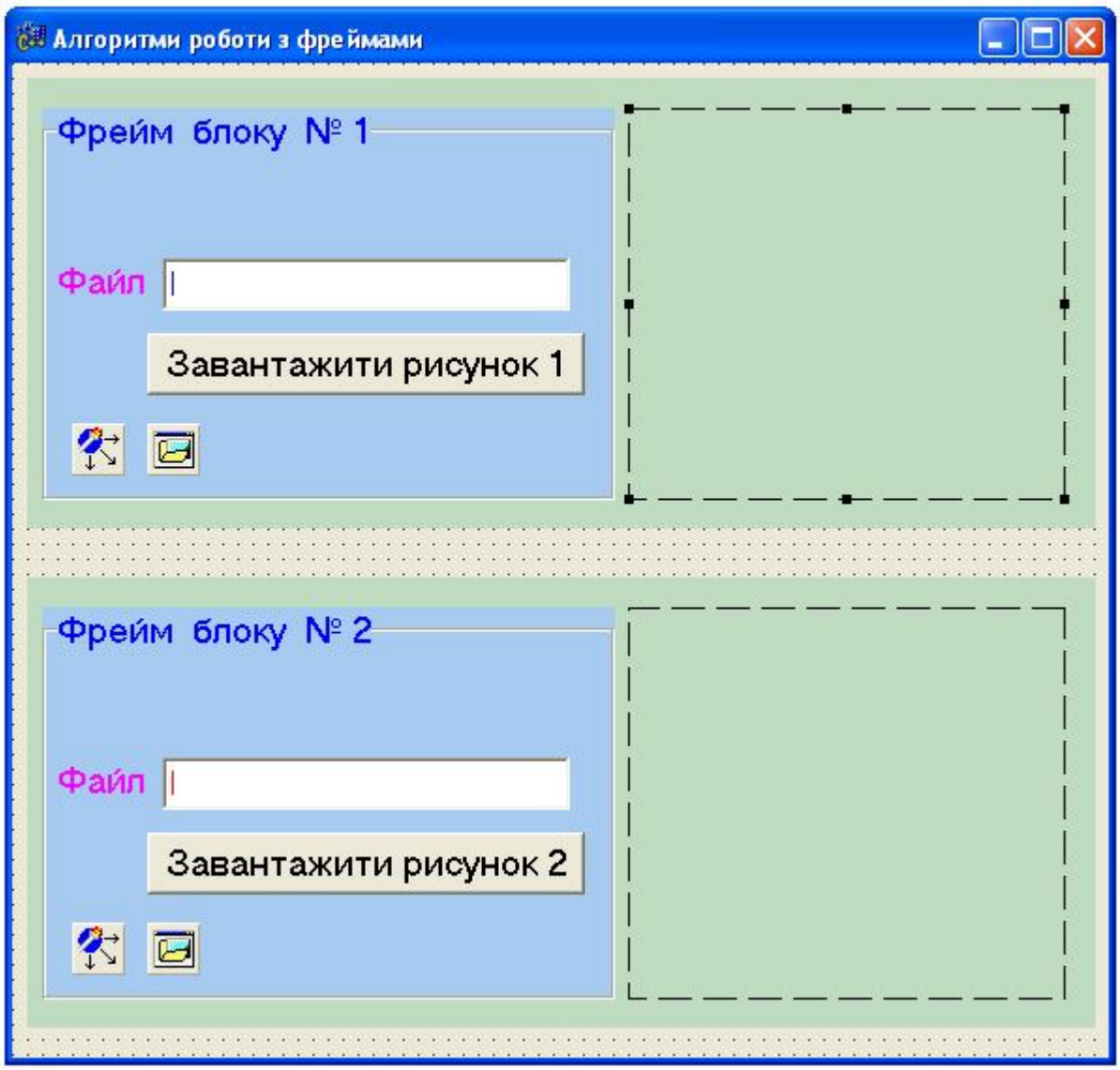

Рис. 4-8. Зображення форми прикладної C++ програми з використанням фреймів.

Крок 7. Далі необхідно для створеного фрейму (рис. 4-6) виконати налагодження кольорів відповідно до рис. 4-7.

Крок 8. Далі необхідно створений фрейм (рис. 4-7) використати для розробки форми вікна прикладної С++ програми. Встановлюємо на форму два рази створений фрейм і вносимо необхідні зміни відповідно до рис. 4-8.

Крок 9. Далі необхідно перейти у вікно створеного фрейму (рис. 4-7), де виконати зміни кольорів у зображенні полів і одночасно спостерігати прояви цих змін у вікні форми. Можна також додати ще одну компоненту в фрейм і далі перевірити появу цей компоненти у вікні форми програми.

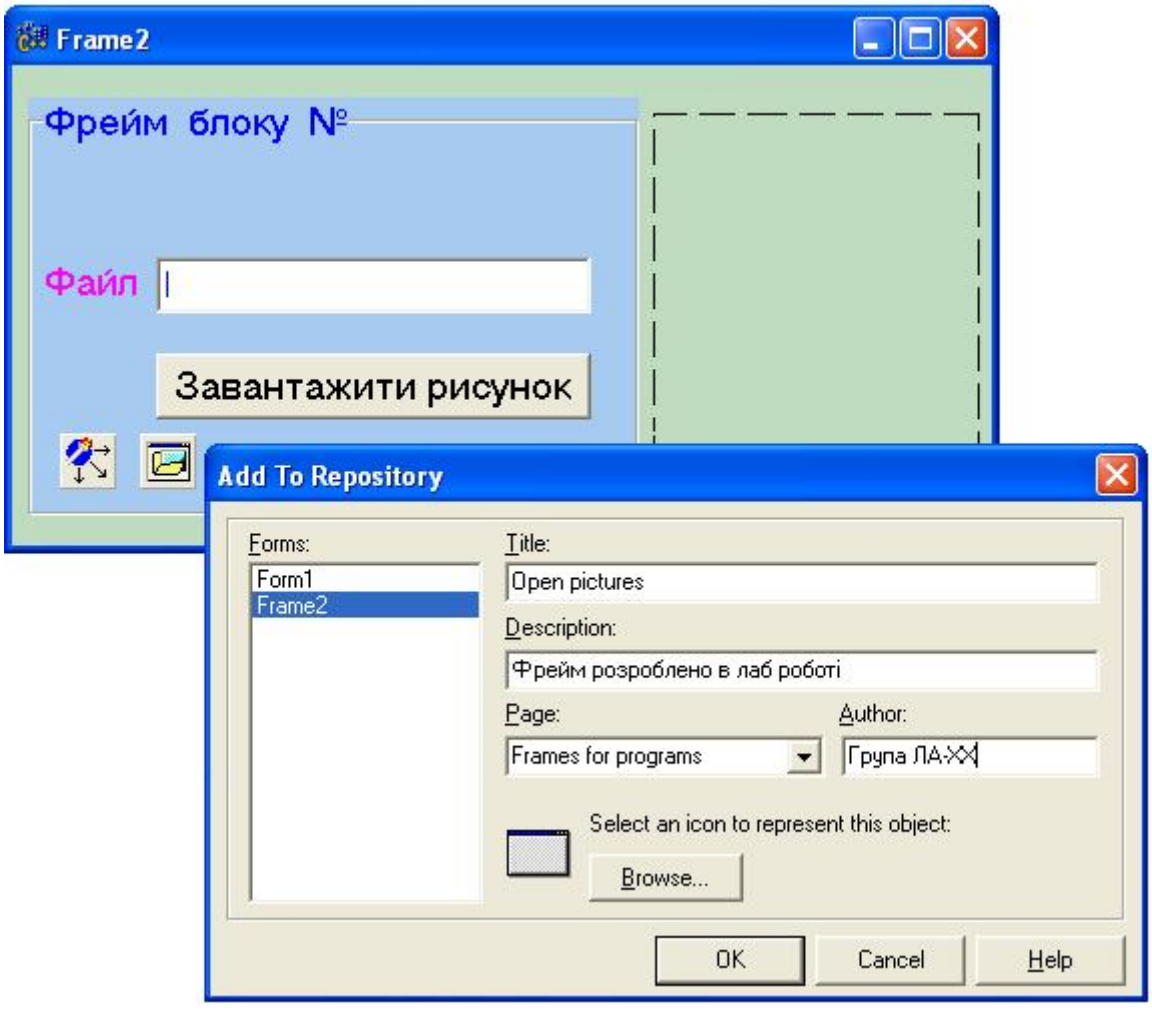

Рис. 4-9. Вікно діалогу при розміщенні фрейму в Депозитарій С++ Builder.

Методика виконання задачі Б.

***Порядок дій і команд при виконанні програмування задачі Б з розробкою нового фрейму завданої структури*:**

Крок 1. Спочатку необхідно розроблений фрейм у задачі А розмістити в Депозитарій ***C++Builder*** і для цього необхідно виконати наступні дії і команди:

- Для цього після щиглика правою кнопкою мишки на фреймі обираємо команду ***Add To Repositary*** для відкриття вікна діалогу та вказуємо відповідні написи (рис. 4-9).

Крок 2. Після розміщення фрейму в Депозитарій необхідно виконати команди ***File / New / Other*** і в результаті відривається діалогове вікно ***New Items***, де можна побачити запис нового фрейму (рис. 4-10), закладку з назвою ***Frames for programs*** та значок іконки з назвою ***Open pictures***.

При виконанні щиглика правою кнопкою мишки на полі закладки відкриється контекстне меню з додатковими командами. Також знизу три радіо-кнопки ***Copy***, ***Inherit*** та ***Us*** встановлюють режим використання фрейму на формі вікна прикладної програми.

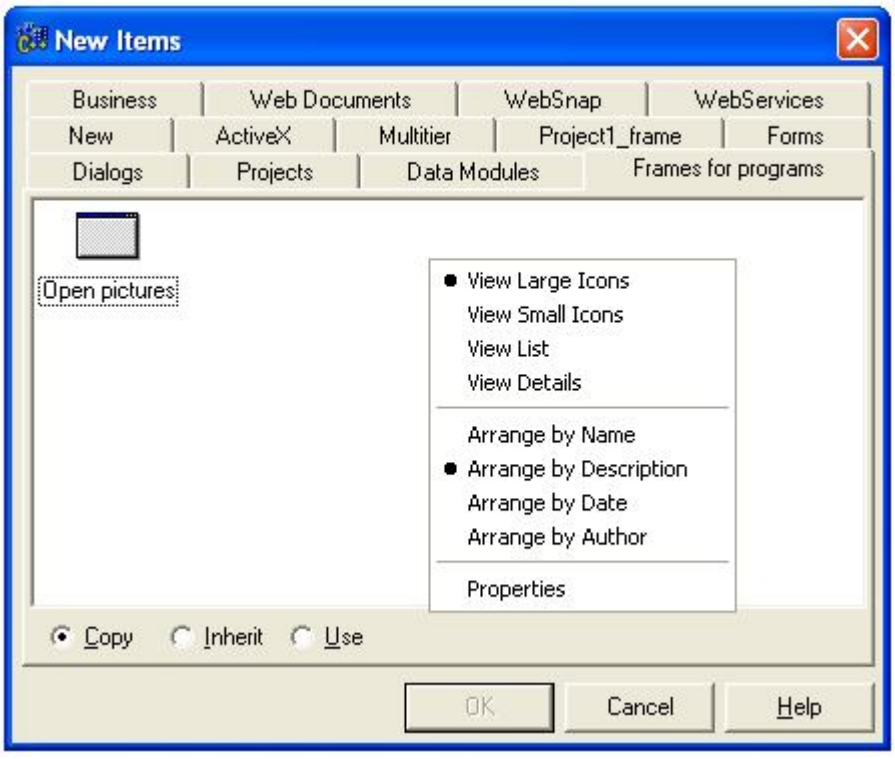

Рис. 4-10. Вікно ***New Items*** з записом нового фрейму в Депозитарій.

Крок 3. Далі необхідно для нового проекту ***Project*** створити три нові чисті форми вікон і три рази на кожну форму окремо встановити фрейм ***Open pictures*** з обранням режиму радіо-кнопки ***Copy***, потім – радіо-кнопки ***Inherit*** та радіо-кнопки ***Us***. Зберігаємо файли нового проекту ***Project***, як проект ***P_Lab4_B*** в папку ***LAB4_B*** з папки ***LAB4*** .

Крок 4. Далі необхідно у фреймі на кожній формі проекту ***P_Lab4_B*** вносити зміни і дослідити, яким чином проявляє себе у фреймі наслідування властивостей. Результати досліджень по кроку 4 показати в протоколі лабораторної роботи № 4 при допомозі скрин-шотів з екрану дисплея комп'ютера.

Контрольні запитання до лабораторної роботи № 4

1. Пояснить блок-схему алгоритму зі створення фрейму.
2. Продемонструйте вплив встановлення режиму ***Copy*** для радіо-кнопки у вікні ***NewItems***.
3. Продемонструйте вплив встановлення режиму ***Inherit*** для радіо-кнопки у вікні ***NewItems***.
4. Продемонструйте вплив встановлення режиму ***Us*** для радіо-кнопки у вікні ***NewItems***.
5. Створіть фрейм для візуалізації графіку з відповідних даних.

Лабораторна робота № 5

Компоненти C++ Builder для візуалізації структури ієрархічних даних на носіях інформації

Ціль виконання лабораторної роботи. Передбачається виконання таких навчальних задач:

- вивченні методики створення прикладної ***C***++ програми для обробки і візуалізації ієрархічних структур даних, зберегаємих на носіях інформації;

- практична робота з компонентами ***C++ Builder,*** забезпечуючих перегляд даних, записаних на дисках комп'ютера, CD-D та інших носіях інформації.

Загальні зведення до компонент зі структур даних

У бібліотеці VCL для відображення ієрархічних структур даних маються такі компоненти.

Таблиця № 5.1.

Компонента	Страница VCL	Опис компоненти
TreeView (вікно дерева даних)	Win32	Перегляд структури ієрархічних даних у вигляді дерева в стилі Windows95/98/2000
OutLine (вікно дерева даних)	Win3.1	Перегляд структури ієрархічних даних у вигляді дерева в стилі Windows 3
ListView (список даних в стилі Windows 95)	Win32	Відображення в стилі папок Windows списком у вигляді піктограм або стовпчиків
OpenDialog (відкрити файл)	Dialogs	Призначений для створення вікна діалогу "Відкрити файл"
FileListBox (список файлів)	Win3.1	Відображає список усіх файлів каталогу
DirectoryListBox (структура каталогів)	Win3.1	Відображає структуру каталогів диска
DriveComboBox (список дисків)	Win3.1	Список доступних дисків, що випадає
CDirectoryOutline (дерево каталогів)	Samples	Компонента для відображення структури каталогів обраного диска

Компонента ***ListView*** дозволяє в стилі ***Windows*** відображати дані у вигляді списків, таблиць, великих і дрібних піктограм. Користувач зіштовхується з таким відображенням даних при розкритті папок у ***Windows***.

Стиль відображення інформації визначається властивістю ***ViewStyle*** , що може встановлюватися в процесі створення прикладної програми або програмно під час виконання. Властивість може приймати такі значення: ***vsicon*** - великі значки; ***vsSmallicon*** - дрібні значки; ***vsList*** - список; ***vsReport*** - таблиця. Основна властивість компонента ***ListView*** задається в полі

властивостей - ***Items***. Реорганізація дерева, зв'язана зі створенням або з видаленням многих вузлів з'єднань, може викликати неприємні мерехтіння зображення. Уникнути цього можна за допомогою методів ***BeginUpdate*** і ***EndUpdate***. Первій метод забороняє перемальовування дерева, а другий - дозволяє.

Постановка задачі по програмуванню у лабораторній роботі

У лабораторній роботі № 5 необхідно:

- створити для ***Windows*** прикладну програму в який у вікні форми буде видна структура папок обраного диска і при відкритті подвійним щигликом на папці повинен з'являтися список файлів записаних в папку.

Завдання для аналізу і обробки результатів роботи запрограмованої C++ програми

Після запуску в роботу ***C***++ програми з лабораторної роботи № 5 для захисту результатів програмування необхідно виконати таке:

- нарисувати блок-схему алгоритму до виконання команд прикладної програми з проекту ***P_Lab*5** лабораторної роботи № 5;

Методика виконання задачі.

Порядок дій і команд при виконанні програмування задачі

Крок 1. Перейдіть на форму програми і у властивості ***Caption*** напишіть назву роботи "Лаб. робота № 5 компонента ***ListView*** ", щоб цей текст з'явився в заголовку форми. Виконаєте команду ***File/Save Project As...*** . На диску ***D***:\ створіть папку Lab_5 для файлів проекту ***P_Lab*5.*bpr*** і файлу ***U_Lab*5.*cpp***.

Крок 2. Установsnm на форму компоненти ***DriveComboBox*1** і ***DriveComboBox*2** , як це показано рис. 5.1.

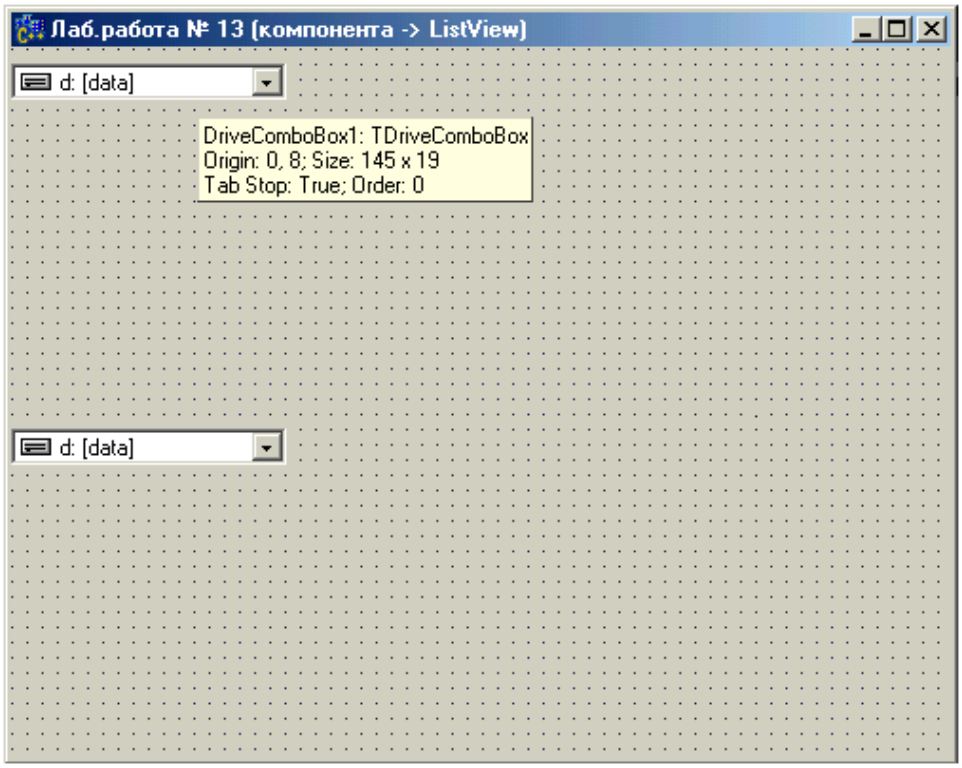

Рис. 5.1.

Крок 3. Установіть на форму компоненти ***DirectoryListBox*1** та ***DirectoryListBox*2 ,** як це показано рис. 5.2.

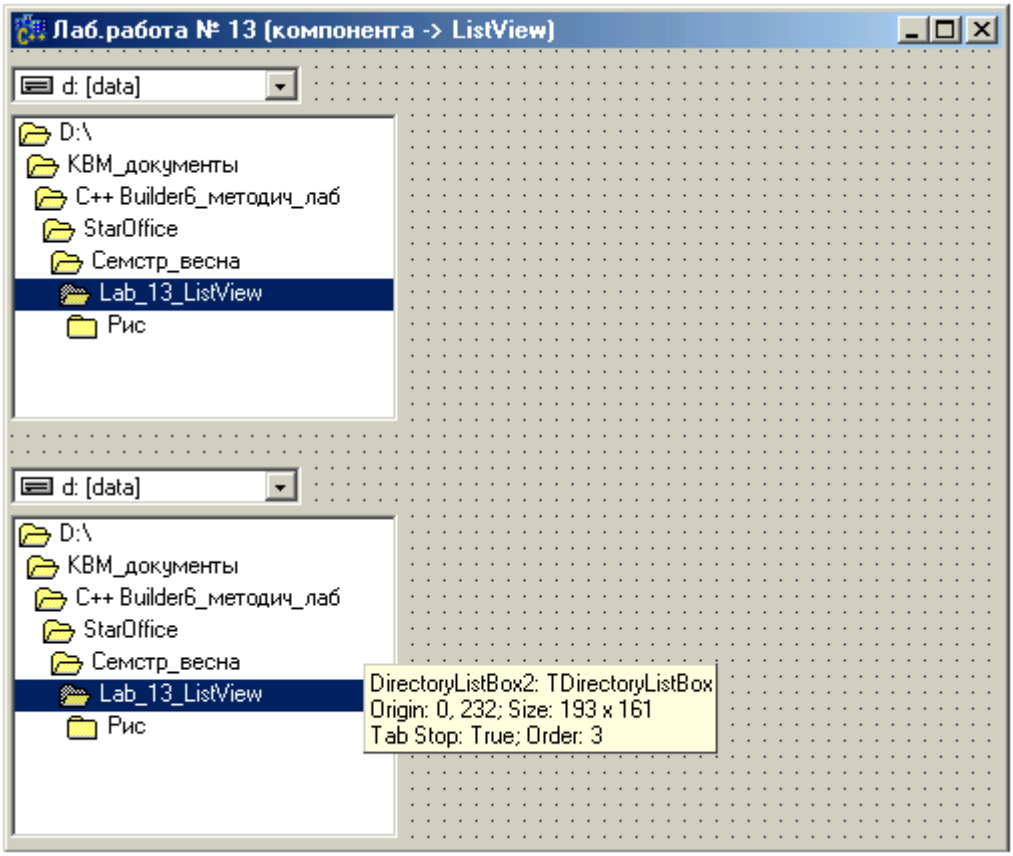

Рис.5.2.

Крок 4. Установіть на форму компонента ListView1 і ListView2 , як це показано рис. 5.3.

Крок 5. Виконаєте попередню компіляцію тексту програми з метою перевірки.

Крок 6. Установіть на форму компоненти ***Label*1** і ***Label*2** . У властивості ***Caption*** задайте синім кольором пояснювальний напис "Властивості значків" і розташуєте їх згідно рис. 5.4.

Крок 7. . Установіть на форму компоненти ***ComboBox*1** та ***ComboBox*2** , як це показано рис. 5.4. У властивості ***Items*** задайте назви для списку, що випадають: **«Великі значки» або «Мали значки» або «Список» чи «Таблиця» .**

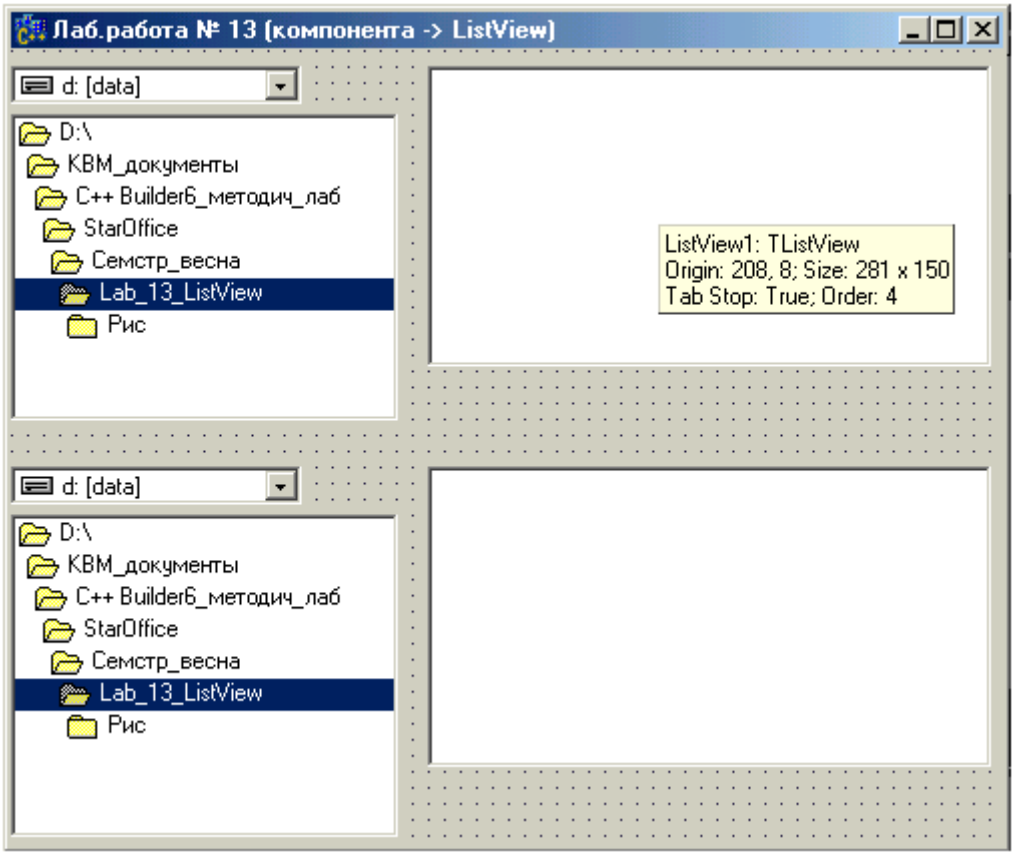

Рис.5. 3.

Крок 8. Задайте подію **void __fastcall TForm1::FormCreate(TObject *Sender);** і заповніть оператори відповідно до лістингу файла ***U_Lab*5.*cpp*** .

Крок 9. Задайте таки 2 події**:**

void __fastcall TForm1::DirectoryListBox1Change(TObject *Sender);

void __fastcall TForm1::DirectoryListBox2Change(TObject *Sender);

і заповніть оператори згідно листингу файлу ***U_Lab5.cpp*** .

Крок 10. Задайте наступні 2 події:

void __fastcall TForm1::ComboBox1Click(TObject *Sender);

void __fastcall TForm1::ComboBox2Click(TObject *Sender);

і заповніть оператори згідно листингу файлу ***U_Lab5.cpp*** .

Крок 11. Виконаєте компіляцію файлів проекту ***P_Lab5.bpr*** та перевірте результат роботи прикладної програми, як показано на рис. 5.5.

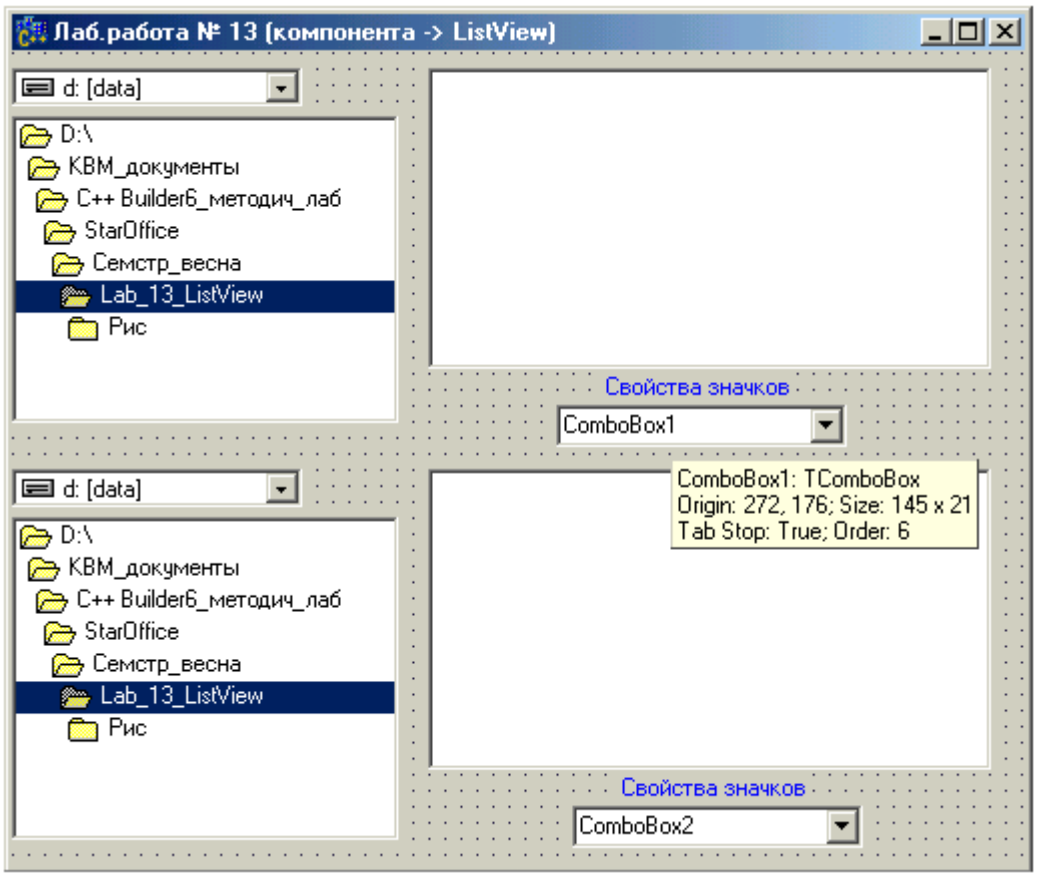

Рис. 5.4.

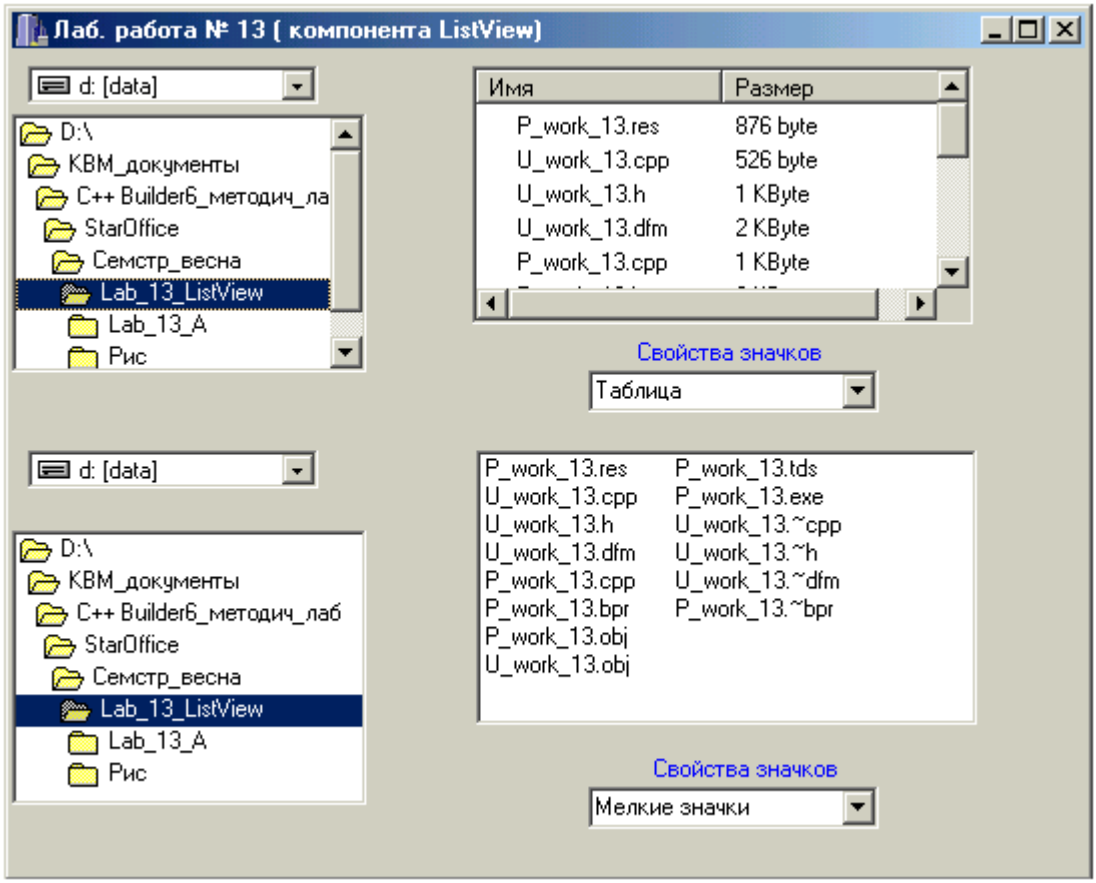

Рис. 5.5. Результат работи програми у ***Windows***.

Листинг файлу ***U_Lab5.cpp***

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "U_Lab5.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void UpdateFiles_1()
{
 TSearchRec sr1;
// TSHFileInfo * fi1;
 TListItem *pItem1 = NULL;
 int IconIndex1;
```

```
 TSHFileInfo * fi1 = new TSHFileInfo;
 Form1->ListView1->Items->BeginUpdate();
 Form1->ListView1->Items->Clear();
 if (FindFirst(Form1->DirectoryListBox1->Directory + "\\*.*", faAnyFile, sr1) == 0)
  do
  {
   if (sr1.Attr == faDirectory) continue;
   pItem1 = Form1->ListView1->Items->Add();
   pItem1->Caption = sr1.Name;
   SHGetFileInfo(("*" + LowerCase(ExtractFileExt(pItem1->Caption))).c_str(),
   0, fi1, sizeof(fi1),
   SHGFI_SMALLICON | SHGFI_SYSICONINDEX | SHGFI_TYPENAME);
   pItem1->ImageIndex = fi1->iIcon;
   if (sr1.Size < 1024)
    pItem1->SubItems->Add(IntToStr(sr1.Size) + " byte");
        else if (sr1.Size < 1024 * 1024)
     pItem1->SubItems->Add(IntToStr(sr1.Size / 1024) + " KByte");
        else pItem1->SubItems->Add(IntToStr(sr1.Size / (1024 * 1024)) + " MByte");
    pItem1->SubItems->Add(fi1->szTypeName);
   }
   while (FindNext(sr1) == 0);
 FindClose(sr1);
 Form1->ListView1->Items->EndUpdate();
}
//---------------------------------------------------------------------------
void UpdateFiles_2()
{
 TSearchRec sr2;
// TSHFileInfo * fi2;
 TListItem *pItem2 = NULL;
 int IconIndex2;
 TSHFileInfo * fi2 = new TSHFileInfo;
 Form1->ListView2->Items->BeginUpdate();
 Form1->ListView2->Items->Clear();
 if (FindFirst(Form1->DirectoryListBox2->Directory + "\\*.*", faAnyFile, sr2) == 0)
  do
  {
   if (sr2.Attr == faDirectory) continue;
   pItem2 = Form1->ListView2->Items->Add();
   pItem2->Caption = sr2.Name;
   SHGetFileInfo(("*" + LowerCase(ExtractFileExt(pItem2->Caption))).c_str(),
     0, fi2, sizeof(fi2),
      SHGFI_SMALLICON | SHGFI_SYSICONINDEX | SHGFI_TYPENAME);
   pItem2->ImageIndex = fi2->iIcon;
   if (sr2.Size < 1024)
    pItem2->SubItems->Add(IntToStr(sr2.Size) + " byte");
        else if (sr2.Size < 1024 * 1024)
     pItem2->SubItems->Add(IntToStr(sr2.Size / 1024) + " KByte");
```

```
        else pItem2->SubItems->Add(IntToStr(sr2.Size / (1024 * 1024)) + " MByte");
    pItem2->SubItems->Add(fi2->szTypeName);
   }
   while (FindNext(sr2) == 0);
 FindClose(sr2);
 Form1->ListView2->Items->EndUpdate();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 TSHFileInfo * fi1, *fi2;
 DriveComboBox1->DirList = DirectoryListBox1;
 ListView1->SmallImages->Height = 16;
 ListView1->SmallImages->Width = 16;
 DriveComboBox2->DirList = DirectoryListBox2;
 ListView2->SmallImages->Height = 16;
 ListView2->SmallImages->Width = 16;
// SHGetFileInfo("*.*", 0, fi,
//      sizeof(fi), SHGFI_SMALLICON | SHGFI_ICON | SHGFI_SYSICONINDEX);
 ListView1->SmallImages->Handle =
 SHGetFileInfo("*.*", 0, fi1,
     sizeof(fi1), SHGFI_SMALLICON | SHGFI_ICON | SHGFI_SYSICONINDEX);
 ListView1->LargeImages->Handle = SHGetFileInfo("*.*", 0, fi1,
     sizeof(fi1), SHGFI_LARGEICON | SHGFI_ICON | SHGFI_SYSICONINDEX);
 ComboBox1->ItemIndex = 0;

 ListView1->SmallImages->Handle =
 SHGetFileInfo("*.*", 0, fi2,
     sizeof(fi2), SHGFI_SMALLICON | SHGFI_ICON | SHGFI_SYSICONINDEX);
 ListView2->LargeImages->Handle = SHGetFileInfo("*.*", 0, fi2,
     sizeof(fi2), SHGFI_LARGEICON | SHGFI_ICON | SHGFI_SYSICONINDEX);
 ComboBox2->ItemIndex = 0;
 UpdateFiles_1();
 UpdateFiles_2();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::DirectoryListBox1Change(TObject *Sender)
{
 UpdateFiles_1();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox1Click(TObject *Sender)
{
 switch (ComboBox1->ItemIndex)
 {
  case 0: ListView1->ViewStyle = vsIcon;
          break;
  case 1: ListView1->ViewStyle = vsSmallIcon;
```

```
        break;
  case 2: ListView1->ViewStyle = vsList;
        break;
  case 3: ListView1->ViewStyle = vsReport;
 }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ComboBox2Click(TObject *Sender)
{
  switch (ComboBox2->ItemIndex)
 {
  case 0: ListView2->ViewStyle = vsIcon;
        break;
  case 1: ListView2->ViewStyle = vsSmallIcon;
        break;
  case 2: ListView2->ViewStyle = vsList;
        break;
  case 3: ListView2->ViewStyle = vsReport;
 }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::DirectoryListBox2Change(TObject *Sender)
{
   UpdateFiles_2();
}
//---------------------------------------------------------------------------
```

Листинг файлу ***U_Lab5.h***

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

#ifndef U_Lab5H
#define U_Lab5H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <FileCtrl.hpp>
#include "ShellAPI.h"
#include "shlwapi.h"
#include <ImgList.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published: // IDE-managed Components
```

```
        TListView *ListView1;
        TDirectoryListBox *DirectoryListBox1;
        TDriveComboBox *DriveComboBox1;
        TComboBox *ComboBox1;
        TImageList *ImageList1;
        TImageList *ImageList2;
        TLabel *Label1;
        TDriveComboBox *DriveComboBox2;
        TDirectoryListBox *DirectoryListBox2;
        TListView *ListView2;
        TLabel *Label2;
        TComboBox *ComboBox2;
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall DirectoryListBox1Change(TObject *Sender);
        void __fastcall ComboBox1Click(TObject *Sender);
        void __fastcall ComboBox2Click(TObject *Sender);
        void __fastcall DirectoryListBox2Change(TObject *Sender);
private:        // User declarations
public:                 // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Контрольні запитання до лабораторної роботи № 5

1. Покажить, як налаштувати у програмі переміщення файлу або папки.
2. Покажить необхідні налаштування у програмі для копіювання файлу.
3. Додайте у програму індикатори для візуального контролю процесу переміщення об’єкта.
4. Додайте у програму індикатори для візуального контролю процесу копіювання файлу.
5. Покажить, яким чином можна у програмі змінювати налаштування розмірів значків.

Література

1. Культин Н.Б. Самоучитель C++ Builder [Текст] /Н.Б. Культин – СПб.: БХВ-Петербург, 2004. – 320с. Библиграф.: с. 318. 4000 экз.
2. Архангельский А.Я. Программирование в C++Builder6 [Текст] /А.Я. Архангельский -М.:ЗАО “Издательство БИНОМ”, 2002. – 1152 с. Библиграф.: с. 1149-1150. 4000 экз.
3. Архангельский А.Я. C++Builder6. Справочное пособие. Книга 1. Язык C++ [Текст] /А.Я. Архангельский – М.: Бином-Пресс, 2002. – 544 с. Библиграф.: с. 543. 4000 экз.
4. Архангельский А.Я. C++Builder6. Справочное пособие. Книга 2. Классы и компоненты [Текст] /А.Я. Архангельский – М.: Бином-Пресс, 2002. – 528 с. Библиграф.: с. 527. 4000 экз.