



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря Сікорського»

ОСНОВИ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ

Частина 2. Основи програмування

Комп'ютерний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Системи керування літальними апаратами та комплексами»
спеціальності 173 «Авіоніка»

Укладач М.М.Чепілко

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2024

УДК 004.43

Укладач: *Чепілко М.М., доктор фіз.-мат.наук, професор*

Рецензент: *Смірнов С.А., канд.фіз.-мат.наук, доцент, навчально - науковий фізико - технічний інститут*

Відповідальний редактор: *Пономаренко С.О., кандидат технічних наук, доцент*

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол №8 від 20.06.2024 р.)

за поданням Вченої ради навчально - наукового інституту аерокосмічних технологій
(протокол №5/24 від 03.06.2024 р.)

”Основи алгоритмізації та програмування. Частина 2. Основи програмування. Комп’ютерний практикум” [Електронний ресурс]: комп’ютерний практикум: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Системи керування літальними апаратами та комплексами» спеціальності 173 «Авіоніка» / КПІ ім. Ігоря Сікорського; укладач: Чепілко М.М.. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 101 с.

Метою видання навчального посібника ”Основи алгоритмізації та програмування. Частина 2. Основи програмування. Комп’ютерний практикум” є допомога студентам у закріпленні та поглибленому розумінні теоретичних положень та методів структурного та об’єктно-орієнтованого програмування на мові C++, набутті досвіду побудови практично значимих програмних додатків, моделюванні та аналізі актуальних фізико-технічних процесів стосовно аерокосмічних технологій шляхом виконання спеціально сформульованих 8-ми комплексних завдань для самостійної роботи. Виконання завдань вимагає активного використання студентом набутих знань з вищої математики, фізики та основ алгоритмізації і програмування.

У кожному завданні розглянуто певний фізико-технічний сценарій, описана його математична модель, наведено приклад екранної форми та програмного коду комп’ютерного додатку, де робиться акцент на тому чи іншому методі структурного чи об’єктно-орієнтованого програмування на мові C++. Надаються методичні рекомендації, що до виконання завдання.

Захист виконаних завдань передбачає аналіз функціоналу відкомпільованого виконавчого файлу та якості написання програмного коду. Кожний підрозділ посібника також містить запитання студентам для самоконтролю їх знань.

УДК 004.43

Реєстр. №XX/XX-XXX.. Обсяг Х.Х авт.арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Берестейський проспект, 37, Київ, 03056
<https://kpi.ua>

Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК №5354 від 25.05.2017 р.

Зміст

1	Комп'ютерний практикум №1. Програмування додатку для розрахунку траєкторій польоту снаряду з врахуванням сили опору повітря	6
1.1	Теоретичні відомості	7
1.2	Рекомендації щодо виконання завдання	9
1.3	Варіанти завдань до комп'ютерного практикуму	14
1.4	Підготовка до комп'ютерного практикуму	15
1.5	Порядок виконання завдання	15
1.6	Зміст звіту щодо виконаної роботи	16
1.7	Контрольні запитання	16
2	Комп'ютерний практикум №2. Програмування додатку для побудови траєкторій польоту ракети в межах земної атмосфери з врахуванням режимів роботи двигуна та сили опору повітря .	17
2.1	Теоретичні відомості	18
2.2	Рекомендації щодо виконання завдання	20
2.3	Варіанти завдань до комп'ютерного практикуму	25
2.4	Підготовка до комп'ютерного практикуму	25
2.5	Порядок виконання завдання	26
2.6	Зміст звіту щодо виконаної роботи	26
2.7	Контрольні запитання	27
3	Комп'ютерний практикум №3. Програмування додатку для побудови траєкторій польоту ракети за межі земної атмосфери з врахуванням стартових умов та режимів роботи двигуна . . .	27
3.1	Теоретичні відомості	28
3.2	Рекомендації щодо виконання завдання	30
3.3	Варіанти завдань до комп'ютерного практикуму	36
3.4	Підготовка до комп'ютерного практикуму	36

3.5	Порядок виконання завдання	37
3.6	Зміст звіту щодо виконаної роботи	38
3.7	Контрольні запитання	38
4	Комп'ютерний практикум №4. Програмування додатку "Калькулятор комплексної алгебри" на основі контейнерного типу даних "struct"	38
4.1	Теоретичні відомості	39
4.2	Рекомендації щодо виконання завдання	41
4.3	Підготовка до комп'ютерного практикуму	46
4.4	Порядок виконання завдання	46
4.5	Зміст звіту щодо виконаної роботи	47
4.6	Контрольні запитання	47
5	Комп'ютерний практикум №5. Програмування додатку для файлової бази даних на основі вкладених структур (тип даних "struct")	48
5.1	Теоретичні відомості	48
5.2	Рекомендації щодо виконання завдання	50
5.3	Завдання до комп'ютерного практикуму	53
5.4	Підготовка до комп'ютерного практикуму	54
5.5	Порядок виконання завдання	54
5.6	Зміст звіту щодо виконаної роботи	55
5.7	Контрольні запитання	56
6	Комп'ютерний практикум №6. Програмування класу для розв'язку задачі Коші для довільної системи диференціальних рівнянь 1-го порядку у випадку немонотонних динамічних систем	56
6.1	Теоретичні відомості	57
6.2	Рекомендації щодо виконання завдання	62
6.3	Варіанти завдань до комп'ютерного практикуму	66
6.4	Підготовка до комп'ютерного практикуму	67
6.5	Порядок виконання завдання	67
6.6	Зміст звіту щодо виконаної роботи	68
6.7	Контрольні запитання	68

7	Комп'ютерний практикум №7. Програмування додатку для дослідження динаміки коливальних систем із дисипативним елементом з використанням алгоритму Адамса-Башфорта	69
7.1	Теоретичні відомості	70
7.2	Рекомендації щодо виконання завдання	73
7.3	Варіанти завдань до комп'ютерного практикуму	77
7.4	Підготовка до комп'ютерного практикуму	77
7.5	Порядок виконання завдання	78
7.6	Зміст звіту щодо виконаної роботи	79
7.7	Контрольні запитання	79
8	Комп'ютерний практикум №8. Програмування додатку "Матрична алгебра (Основні операції матричної алгебри)" на основі об'єктного підходу та дослідження його функціоналу	80
8.1	Теоретичні відомості	80
8.2	Рекомендації щодо виконання завдання	80
8.3	Підготовка до комп'ютерного практикуму	98
8.4	Порядок виконання завдання	98
8.5	Зміст звіту щодо виконаної роботи	100
8.6	Контрольні запитання	100
9	Список використаної літератури	101

Вступ

Самостійне виконання студентами завдань для комп'ютерного практикуму з кредитного модуля "Основи алгоритмізації та програмування. Частина 2. Основи програмування. Комп'ютерний практикум" грає ключову роль у закріпленні та поглибленому розумінні теоретичних положень та методів структурного та об'єктно-орієнтовано програмування на мові C++ [1]-[4], набутті досвіду побудови практично значимих програмних додатків, моделюванні та аналізі актуальних фізико-технічних процесів стосовно аерокосмічних технологій. Виконання завдань комп'ютерного практикуму з кредитного модуля "Основи алгоритмізації та програмування. Частина 2. Основи програмування. Комп'ютерний практикум" вимагає активного використання студентом набутих знань з вищої математика, фізики та основ алгоритмізації і програмування.

Цикл робіт для комп'ютерного практикуму передбачає виконання 8-ми завдань із застосуванням методів структурного та об'єктно-орієнтовано програмування на мові C++. Ритм виконання завдань синхронізовано з освоєнням студентами теоретичного матеріалу та набуттям практичних умінь стосовно об'єктно-орієнтовано програмування, що розглядається під час лекцій та практичних занять.

Завдання для комп'ютерного практикуму доцільно виконувати у програмному середовищі [Embarcadero C++ Builder Community Edition](#) [5] або [Borland C++ Builder Enterprise v6.0](#) [6], в залежності від потужності комп'ютера.

У кожному завданні розглянуто певний фізико-технічний сценарій, описана його математична модель, наведено приклад екранної форми та програмного коду комп'ютерного додатку, де робиться акцент на тому чи іншому методі структурного чи об'єктно-орієнтовано програмування на мові C++. Надаються методичні рекомендації, що до виконання завдання.

Захист виконаних завдань передбачає аналіз функціоналу відкомпільованого виконавчого файлу та якості написання програмного коду.

1. Комп'ютерний практикум №1. Програмування додатку для розрахунку траєкторій польоту снаряду з врахуванням сили опору повітря

Мета роботи: Отримати досвід задавання таких контейнерних типів даних мови програмування C++, як `<vector>` і `<iterator>` та використання їх у практичному програмуванні.

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] візуальний додаток для побудови траєкторій польоту снаряду з врахуванням сили опору повітря у квадратичному наближенні [7] з використанням контейнерних типів даних `<vector>` і `<iterator>` для наступних початкових умов: у момент $t = 0$ $\vec{r}(t = 0) = \vec{r}_0$, $\vec{v}(t = 0) = \vec{v}_0$, де $\vec{r} = (r_x, r_y, r_z)$, $\vec{v} = (v_x, v_y, v_z)$ — координати та швидкість руху снаряду масою m . Сила опору повітря для снаряда визначається формулою

$$\vec{f}_{fr} = -(k_1 + k_2|\vec{v}|)\vec{v}, \quad |\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2},$$

де k_1, k_2 — коефіцієнти тертя 1-го та 2-го полядку.

Вхідні параметрами задачі:

- Координати гармати
- Інертна маса снаряду
- Початкова швидкість снаряду
- Лінійний та квадратичний коефіцієнти тертя

Розрахункові параметрами задачі:

- Час досягання екстремуму траєкторії польоту снаряду
- Максимальна висота підйому снаряду
- Горизонтальне зміщення снаряду в момент падіння на землю
- Швидкість снаряду в момент падіння на землю
- Кінетична енергія снаряду в момент падіння на землю

1.1. Теоретичні відомості

Політ снаряду визначається 2-м законом Ньютона [7]

$$m \frac{d^2 \vec{r}}{dt^2} = m \vec{g} + \vec{f}_{fr}$$

або

$$\begin{cases} m \frac{d\vec{v}}{dt} = m \vec{g} + \vec{f}_{fr} \\ \frac{d\vec{r}}{dt} = \vec{v} \end{cases}$$

Для подальших розрахунків раціонально перейти до безрозмірних змінних, а саме

$$\tau = \frac{t}{t_0}, \quad \vec{V} = \frac{\vec{v}}{|\vec{v}_0|}, \quad \vec{R} = \frac{\vec{r}}{r_0},$$

де

$$t_0 = \frac{|\vec{v}_0|}{g}, \quad r_0 = v_0 t_0.$$

В нових змінних рівняння руху снаряду набудуть вигляд

$$\begin{cases} \frac{d\vec{V}}{d\tau} = \frac{\vec{g}}{g} - (\beta_1 + \beta_2 |\vec{V}|) \vec{V} \\ \frac{d\vec{R}}{d\tau} = \vec{V} \end{cases}$$

де $\beta_1 = k_1 \frac{t_0}{m}$, $\beta_2 = k_2 \frac{r_0}{m}$. Ці рівняння являються нелінійними. Тому їх розв'язки можна отримати лише методами обчислювальної математики.

Зазвичай для розв'язку таких систем рівнянь використовують метод Рунге-Кути 4-го порядку [2]. Схема використання цього методу наступна. Нехай

$$\frac{d\vec{r}}{dt} = \vec{f}(t, \vec{r}), \quad \vec{r}(t_0) = \vec{r}_0, \quad t, \Delta t \in R^1, \quad \vec{r}, \vec{f}, \vec{k}_p^i \in R^n.$$

Тоді послідовність наближених значень $\vec{r}_i = \vec{r}(t_i)$ починаючи з $\vec{r}(t_0) = \vec{r}_0$ обчислюються за наближеною ітераційною формулою

$$\vec{r}_{i+1} = \vec{r}_i + \frac{\Delta t}{6} (\vec{k}_1^i + 2\vec{k}_2^i + 2\vec{k}_3^i + \vec{k}_4^i)$$

$$\begin{cases} \vec{k}_1^i = \vec{f}(t_i, \vec{r}_i), \\ \vec{k}_2^i = \vec{f}\left(t_i + \frac{\Delta t}{2}, \vec{r}_i + \frac{\Delta t}{2} \vec{k}_1^i\right), \\ \vec{k}_3^i = \vec{f}\left(t_i + \frac{\Delta t}{2}, \vec{r}_i + \frac{\Delta t}{2} \vec{k}_2^i\right), \\ \vec{k}_4^i = \vec{f}\left(t_i + \Delta t, \vec{r}_i + \vec{k}_3^i\right), \end{cases}$$

де Δt — крок зміщення вздовж часової осі t . Сумарна помилка обчислень при цьому складає $O(\Delta t^4)$.

Приклад програмного коду, що реалізує поставлену задачу, приведено нижче. Оскільки кількість ітерацій при використанні методу розв'язку системи диференціальних рівнянь Рунге-Кути наперед невідома, то виглядає раціональним використання в даному випадку контейнерного типу даних **vector**, який дозволяє формувати універсальний код програми та динамічно

нарощувати кількість компонент в ітеративних процесах. При цьому доцільно програмний код методу розв'язку системи диференціальних рівнянь Рунге-Кути винести в окремий універсальний заголовочний файл **RK4.h**, який можна буде використовувати і в майбутніх проектах.

```
/*-----*/
/* Заголовочний файл RK4.h */
/*-----*/
vector<float> fn_RK4(vector<float> (*F)(vector<float> X),
vector<float> U, float dt) {
int N = U.size();
float t = U[0];
vector<float> K1(N),K2(N),K3(N),K4(N);
K1 = F(U);
U[0] = t+dt/2;
for(int i=1; i<=N; i++) {U[i] = U[i]+dt*K1[i]/2;}
K2 = F(U);
U[0] = t+dt/2;
for(int i=1; i<=N; i++) {U[i] = U[i]+dt*K2[i]/2;}
K3 = F(U);
U[0] = t+dt;
for(int i=1; i<=N; i++) {U[i] = U[i]+dt*K3[i];}
K4 = F(U);
for(int i=1; i<=N; i++) {
U[i] = U[i]+dt*(K1[i]+K2[i]+K3[i]+K4[i])/6;}
return U;
}
/*-----*/
```

Ключовою властивістю контейнерного типу даних **vector** є те, що його можна використовувати (наряду із звичайними змінними) в якості аргументів функцій та їх значень. Особливістю коду функції **fn_RK4** є те що в числі її аргументів є показник в функції **vector<float> (*F)(vector<float> X)** та її vector-аргумент **vector<float> U**. За рахунок цих особливостей код функції **fn_RK4** є універсальним і може використовуватися в інших проектах без жодних змін.

1.2. Рекомендації щодо виконання завдання

Приклад головної форми додатку приведено Рис.1.1. Там же приведені вхідні дані задачі, що задаються по замовчуванні. Для її побудови необхідно

використати такі компоненти

$$\begin{aligned}
 TForm &- 1 \text{ шт.}; & TPanel &- 2 \text{ шт.}; \\
 TGroupBox &- 2 \text{ шт.}; & TEdit &- 13 \text{ шт.}; \\
 TLabel &- 13 \text{ шт.}; & TButton &- 3 \text{ шт.}; \\
 TImage &- 1 \text{ шт.}; & TTimer &- 1 \text{ шт.};
 \end{aligned}
 \tag{1.1}$$

які знаходяться на інструментальній панелі.

В головному файлі проекту знаходиться програмний код, який реалізує інтерфейс додатку (див. Рис.1.1) та його функціонал.

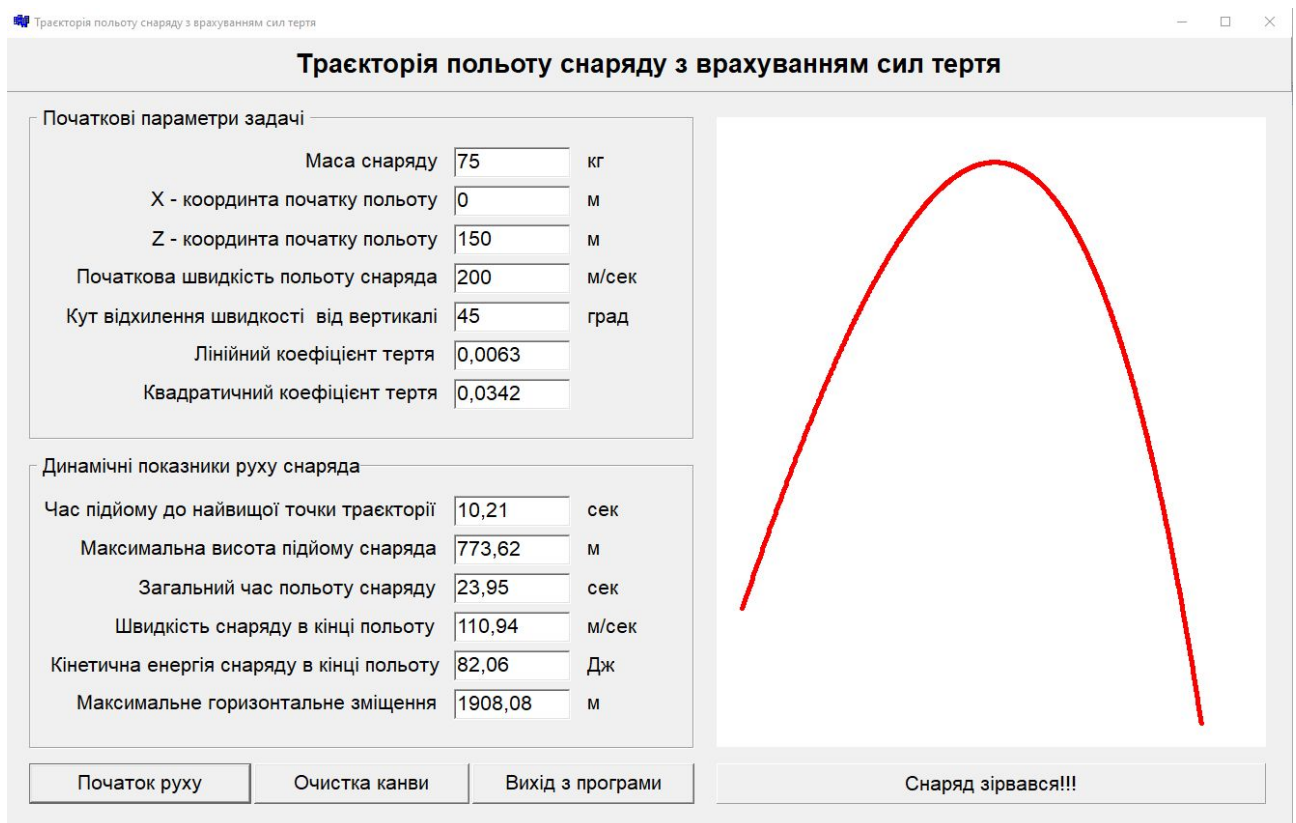


Рис. 1.1. Візуальний додаток для розрахунку траєкторії руху снаряду

Приклад програмного коду головного файлу, що реалізує інтерфейс та функціонал додатку, можна написати у такому вигляді

```

/*- Unit01.cpp -----*/
#include <vcl.h>
#pragma hdrstop
#include <vector>
#include <Math.hpp>
#include <math.h>
using namespace std;

#include "Unit01.h"

```

```

#include "RK4.h"
/*-----*/
#pragma package(smart_init)
#pragma resource "*.dfm"

TfrmUnit01 *frmUnit01;
const float pi = 3.141563;
const float g = 9.8;
int i,Cx,Cz;
float m,Yz,V0,B1,B2,T0,R0,X0,Z0,dt,dX,dZ;
vector<float> U0(5),U(5),Ux(5),Uh(5);
TRect aRect;

    vector<float> fn_Force0(vector<float> U) {
        int N = U.size();
        vector<float> F(N);
        F[0] = 1;
        F[1] = -(B1+B2*sqrt(U[1]*U[1]+U[2]*U[2]))*U[1];
        F[2] = -(B1+B2*sqrt(U[1]*U[1]+U[2]*U[2]))*U[2]-1;
        F[3] = U[1];
        F[4] = U[2];
        return F;
    }

/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner)
    : TForm(Owner)
{}

/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)
{
    try {
        frmUnit01->Caption = frmUnit01->Header0->Caption;
        Timer0->Enabled = false;
        dt = 0.01;
        m = StrToFloat(Edit0->Text);
        V0 = StrToFloat(Edit3->Text);
        T0 = V0/g;
        R0 = V0*T0;
        dt = dt/T0;
        Yz = pi*StrToFloat(Edit4->Text)/180;
        B1 = StrToFloat(Edit5->Text)*T0/StrToFloat(Edit0->Text);
        B2 = StrToFloat(Edit6->Text)*R0/StrToFloat(Edit0->Text);
    }
}

```

```

X0 = StrToFloat(Edit1->Text)/R0;
Z0 = StrToFloat(Edit2->Text)/R0;

U0[0] = 0;
U0[1] = sin(Yz);
U0[2] = cos(Yz);
U0[3] = X0;
U0[4] = Z0;

Button1Click((TObject *)Sender);
} catch(...) {
    ShowMessage("Помилка введення даних!");
}}
/*-----*/
void main(){
}
/*-----*/

void __fastcall TfrmUnit01::Button0Click(TObject *Sender)
{
    Edit7->Text = FloatToStr(0);
    Edit8->Text = FloatToStr(0);
    Edit9->Text = FloatToStr(0);
    Edit10->Text = FloatToStr(0);
    Edit11->Text = FloatToStr(0);
    Edit12->Text = FloatToStr(0);
    FormShow((TObject *)Sender);
    Button1Click((TObject *)Sender);
    i = 0;
    Timer0->Enabled = true;
    Panel2->Caption = "Снаряд в польоті";
}
/*-----*/
void __fastcall TfrmUnit01::Timer0Timer(TObject *Sender)
{
    fn_RK4exec0((TObject *)Sender);
    fn_RK4data0((TObject *)Sender);
    fn_RK4image0((TObject *)Sender);
    if (U[4] < 0) {
        Timer0->Enabled = false;
        Panel2->Caption = "Снаряд зірвався!!!";
    }
}

```

```

        };
    }
/*-----*/
void __fastcall TfrmUnit01::fn_RK4exec0(TObject *Sender)
{
    if (i == 0) { U = fn_RK4(fn_Force0, U0, dt); }
    Ux = U;
    U = fn_RK4(fn_Force0, U, dt);
    if (Ux[2]*U[2] < 0) { Uh = Ux; };
    i++;
}
/*-----*/
void __fastcall TfrmUnit01::fn_RK4data0(TObject *Sender) {
float T, H;
if (U[2] > 0) {T = 0; H = 0;} else {T = Uh[0]*T0; H = Uh[4]*R0;}
Edit7->Text = FloatToStr(SimpleRoundTo(T,-2));
Edit8->Text = FloatToStr(SimpleRoundTo(H,-2));
Edit9->Text = FloatToStr(SimpleRoundTo(U[0]*T0,-2));
Edit10->Text =
FloatToStr(SimpleRoundTo(sqrt(U[1]*U[1]+U[2]*U[2])*V0,-2));
Edit11->Text =
FloatToStr(SimpleRoundTo((U[1]*U[1]+U[2]*U[2])*V0*V0/2/m,-2));
Edit12->Text = FloatToStr(SimpleRoundTo(U[3]*R0,-2));
}
/*-----*/
void __fastcall TfrmUnit01::fn_RK4image0(TObject *Sender) {
    dX = U[3]*R0*0.25;
    dZ = -U[4]*R0*0.75;
    if (i <= 1) {Image0->Canvas->Pen->Color=clWhite;}
    else {Image0->Canvas->Pen->Color=clRed;}
    Image0->Canvas->LineTo(Cx+dX,Cz+dZ);
}
/*-----*/
void __fastcall TfrmUnit01::Button1Click(TObject *Sender)
{
    Cx=25;
    Cz=Image0->Height-25;
    Image0->Canvas->Pen->Width=5;
    Image0->Canvas->Pen->Color=clWhite;
    aRect = Rect(0,0,ClientWidth,ClientHeight);
    Image0->Canvas->FillRect(aRect);
}

```

```

Image0->Canvas->FillRect(Rect(0,0,Image0->Width,Image0->Height));
Image0->Canvas->MoveTo(Cx,Cz);
Panel2->Caption = "Пушка заряджена!";
}
/*-----*/
void __fastcall TfrmUnit01::Button2Click(TObject *Sender)
{
frmUnit01->Close();
}
/*-----*/

```

В описовій частині коду тут використовується контейнерний тип даних, а саме 5-вимірний **vector**-функція узагальнених сил **vector<float> fn_Force0(vector<float> U)** аргументом якої є 5-вимірний **vector** узагальнених координат **vector<float> U**. У випадку руху снаряду в площині $X, 0, Z$ до них відносяться t час, V_x, V_z — компоненти швидкості, R_x, R_z — компоненти координат снаряду.

У тілі функції **FormShow(TObject *Sender)** визначаються початкові умови для руху снаряду, функція **Button0Click(TObject *Sender)** визначає початкову фазу руху снаряду, функція **Timer0Timer(TObject *Sender)** запускає ітерації методу Рунге-Кути. В тілі цієї функції використовується допоміжні функції

- **fn_RK4exec0((TObject *)Sender),**
- **fn_RK4data0((TObject *)Sender),**
- **fn_RK4image0((TObject *)Sender).**

Перша з них власне запускає ітераційний процес Рунге-Кути, друга на кожному кроці ітеративного процесу заповнює даними компоненти **Edit** на панелі "Динамічні показники руху снаряда", а третя будує траєкторію польоту снаряду на кожному кроці ітерації.

Перемикання режиму роботи додатку визначається компонентами **TButton**, "натискання" на які викликає на виконання асоційовані з ними функції.

1.3. Варіанти завдань до комп'ютерного практикуму

- Координати гармати: [0,175; 0,180; 0,185; 0,190; 0,195;] м
- Інертна маса снаряду: [75; 80; 85; 90; 95;] кг
- Початкова швидкість снаряду: [200; 225; 250; 255; 260;] м/сек

- Лінійний та квадратичний коефіцієнти тертя: [0.0075,0.0343; 0.0067,0.0345; 0.0085,0.0242; 0.0066,0.0442; 0.0035,0.0542;]

1.4. Підготовка до комп'ютерного практикуму

- Вивчити теоретичний матеріал щодо контейнерних типів даних `<vector>` і `<iterator>`.
- Ознайомитися з теоретичними відомостями щодо теми комп'ютерного практикуму.
- Ознайомитися з рекомендаціями щодо виконання завдання.

1.5. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.
- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project01.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project01.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям до виконання роботи, аналогічно Рис.1.1, розташуйте на головній формі проекту колекцію компонентів 1.1, та надайте їх відповідне оформлення.
- В папці з файлами проекту створіть заголовочний файл **RK4.h** та наповніть його кодом, який реалізує метод Рунге-Кути 4-го порядку (див. Рекомендації щодо ...). У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <vector>`, `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "RK4.h"`.
- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при

написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).

- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Виконайте тестові розрахунки на прикладах з відомою відповіддю.

1.6. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості.

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочного файлу **RK4.h** та головного файлу **Unit01.cpp**;
 - висновки.

1.7. Контрольні запитання

- В чому заключаються переваги використання заголовочного файлу користувача **RK4.h**?
- Як означається контейнерний тип даних `<vector>`?
- У чому полягає відмінність контейнерного типу даних `<vector>` від динамічних масивів?
- Як здійснюється доступ до елементів контейнерного типу даних `<vector>`?
- Як визначається обсяг пам'яті, потрібний для зберігання контейнерного типу даних `<vector>`?
- Чи дозволяється використання контейнерного типу даних `<vector>` в якості типу даних для елементів контейнерного типу даних `<vector>`?

2. Комп'ютерний практикум №2. Програмування додатку для побудови траєкторій польоту ракети в межах земної атмосфери з врахуванням режимів роботи двигуна та сили опору повітря

Мета роботи: Отримати поглиблений досвід використання контейнерних типів даних мови програмування C++ у практичному програмуванні.

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] візуальний додаток для побудови траєкторій польоту ракети в межах земної атмосфери з врахуванням режимів роботи двигуна та сили опору повітря у квадратичному наближенні [7] з використанням контейнерних типів даних для наступних початкових умов: у момент $t = 0$ $\vec{r}(t = 0) = \vec{r}_0$, $\vec{v}(t = 0) = \vec{v}_0$, де $\vec{r} = (r_x, r_y, r_z)$, $\vec{v} = (v_x, v_y, v_z)$ — координати та швидкість руху ракети із змінною масою m . Сила опору повітря для снаряда визначається формулою

$$\vec{f}_{fr} = -(k_1 + k_2|\vec{v}|)\vec{v}, \quad |\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2},$$

де k_1, k_2 — коефіцієнти тертя 1-го та 2-го порядку.

Вхідні параметрами задачі:

- Координати стартового столу
- Маса ракети без палива
- Коефіцієнт Ціолковського
- Кут відхилення початкової швидкості ракети від вертикалі
- Час роботи двигуна ракети
- Відносна швидкість витікання гарячих газів
- Лінійний та квадратичний коефіцієнти тертя

Розрахункові параметрами задачі:

- Час досягання екстремуму траєкторії польоту ракети
- Максимальна висота підйому ракети
- Загальний час польоту ракети
- Горизонтальне зміщення ракети в кінці польоту
- Швидкість ракети в кінці польоту
- Кінетична енергія ракети в кінці польоту

2.1. Теоретичні відомості

Політ ракети з врахуванням сил опору повітря та роботи її двигуна визначається 2-м законом Ньютона [7]

$$m \frac{d^2 \vec{r}}{dt^2} = m \vec{g} + \vec{f}_{fr} + \vec{f}_p$$

або

$$\begin{aligned} m \frac{d\vec{v}}{dt} &= m \vec{g} + \vec{f}_{fr} + \vec{f}_p \\ \frac{d\vec{r}}{dt} &= \vec{v} \end{aligned} \quad (2.1)$$

Сила опору повітря для ракети визначається формулою [7]

$$\vec{f}_{fr} = -(k_1 + k_2 |\vec{v}|) \vec{v}, \quad |\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2},$$

де k_1, k_2 — коефіцієнти опору повітря 1-го та 2-го порядку, а реактивна сила Мещерського $\vec{f}_p$ формулою $\vec{f}_p = \vec{v}_p \frac{dm}{dt}$ [7]. Тут m — маса ракети на траєкторії польоту, а $\vec{v}_p$ — відносна швидкість витікання гарячих газів, яка вважається постійною за величиною.

Для подальших розрахунків раціонально перейти до безрозмірних змінних, а саме

$$\tau = \frac{t}{t_0}, \quad \vec{V} = \frac{\vec{v}}{v_0}, \quad \vec{R} = \frac{\vec{r}}{r_0},$$

де

$$v_0 = g t_0, \quad r_0 = \frac{g t_0^2}{2}.$$

Тут параметр t_0 має зміст проміжку часу на протязі якого працював двигун ракети. В нових змінних рівняння руху ракети набудуть вигляд

$$\begin{aligned} \frac{d\vec{V}}{d\tau} &= \frac{\vec{g}}{g} - \frac{1}{\mu} (\beta_1 + \beta_2 |\vec{V}|) \vec{V} + \vec{F} \\ \frac{d\vec{R}}{d\tau} &= \vec{V}, \end{aligned} \quad (2.2)$$

де $\vec{F} = -\frac{\vec{V}}{|\vec{V}|} \frac{V_p}{\mu} \frac{d\mu}{d\tau}$ — безрозмірна сила Мещерського, $\beta_1 = k_1 \frac{t_0}{m_f}$, $\beta_2 = 2k_2 \frac{r_0}{m_f}$,

$m_f = m_0 + m_1$ — повна маса заправленої ракети на старті, $\mu = \frac{m}{m_f}$ —

безрозмірна маса ракети на траєкторії польоту, $V_p = \frac{v_p}{v_0}$.

Параметр відповідальний за вигорання палива α можна знайти з рівняння

$$\frac{m_0}{m_0 + m_1} = \mu_1, \quad \mu_1 = \mu|_{\tau=1} \quad \text{або} \quad \frac{1}{1 + \chi} = \mu_1,$$

де $\chi = \frac{m_1}{m_0}$ — так названий коефіцієнт Ціолковського.

Роль сили Мещерського $\vec{F}$ визначається режимом роботи двигуна ракети.

1. У випадку коли реактивна сила є постійною величиною темп витікання гарячих газів визначається формулою

$$m = m_f \mu(t) = m_f(1 - \alpha t) = m_f(1 - \alpha_1 \tau), \quad \alpha_1 = \alpha t_0, \quad \mu_1 = (1 - \alpha_1).$$

В цьому випадку отримаємо $\alpha = \frac{1}{t_0} \frac{\chi}{1 + \chi}$. Оскільки параметр задачі χ є власною характеристикою ракети (постійна величина), то реактивну силу $\vec{F}$ в цьому випадку доцільно виразити наступним чином

$$\vec{F} = \alpha_1 \frac{\vec{V}}{|\vec{V}|} \frac{V_p}{\mu}.$$

2. У випадку коли ракета рухається з постійним прискоренням темп витікання гарячих газів визначається формулою

$$m = m_f \mu(t) = m_f \exp(-\alpha t) = m_f \exp(-\alpha_1 \tau), \quad \alpha_1 = \alpha t_0, \quad \mu_1 = \exp(-\alpha_1).$$

В цьому випадку отримаємо $\alpha = \frac{1}{t_0} \ln(1 + \chi)$, а формула для реактивної сили $\vec{F}$ тут набуває вигляду

$$\vec{F} = \alpha_1 \frac{\vec{V}}{|\vec{V}|} V_p.$$

Отримані рівня, що описують рух ракети з врахуванням опору повітря, являються нелінійними. Тому їх розв'язки можна отримати лише методами обчислювальної математики.

В програмній частині завдання є деякі моменти, які вже були розглянуті під час комп'ютерного практикуму №1. В даній роботі також буде потрібно використати заголовочний файл користувача **RK4.h** для чисельного розв'язку задачі Коші (2.2), підключивши його за допомогою інструкції `#include "RK4.h"` до головного файлу проекту.

2.2. Рекомендації щодо виконання завдання

Приклад головної форми додатку приведено Рис.2.1. Там же приведені вхідні дані задачі, що задаються по замовчуванню. Для її побудови необхідно використати такі компоненти

$$\begin{aligned}
 TForm & - 1 \text{ шт.}; & TPanel & - 2 \text{ шт.}; \\
 TRadioButton & - 2 \text{ шт.}; & TGroupBox & - 2 \text{ шт.}; \\
 TEdit & - 13 \text{ шт.}; & TLabel & - 13 \text{ шт.}; & TButton & - 3 \text{ шт.}; \\
 TImage & - 1 \text{ шт.}; & TTimer & - 1 \text{ шт.};
 \end{aligned}
 \tag{2.3}$$

які знаходяться на інструментальній панелі.

В головному файлі проекту знаходиться програмний код, який реалізує інтерфейс додатку (див. Рис.2.1) та його функціонал.

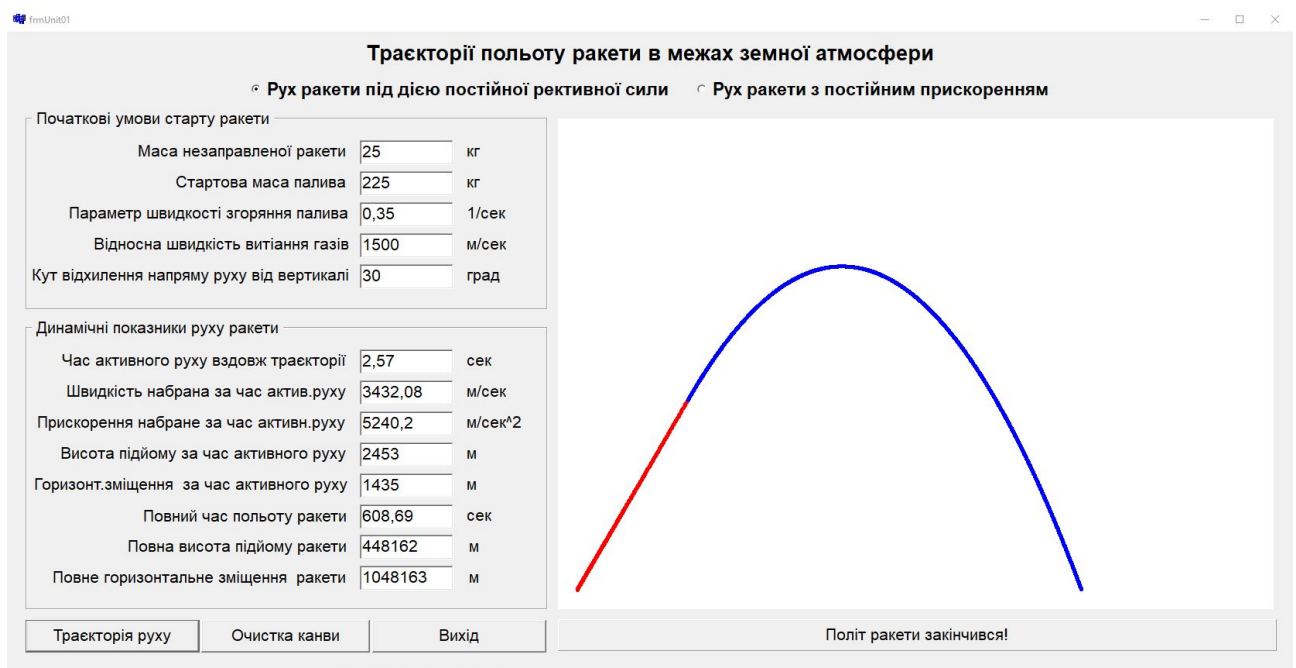


Рис. 2.1. Візуальний додаток для розрахунку траєкторії руху ракети

Приклад програмного коду головного файлу, що реалізує інтерфейс та функціонал додатку, можна написати у такому вигляді

```

/*- Unit01.cpp -----*/
#include <vcl.h>
#pragma hdrstop
#include <vector>
#include <Math.hpp>
#include <math.h>
using namespace std;

#include "Unit01.h"

```

```

#include "RK4.h"
/*-----*/
#pragma package(smart_init)
#pragma resource "*.dfm"

TfrmUnit01 *frmUnit01;
const float pi = 3.141563;
const float g = 9.8;
int flag,i,Cx,Cz;
float M,M0,Mf,Kc,Ka,Yz,T0,V0,R0,B1,B2,V1,T1,X0,Z0,Fp,dt,dX,dZ;
vector<float> U0(5),U(5),Ux(5),Uh(5);
TRect aRect;

    vector<float> fn_Force0(vector<float> U) {
int N = U.size();
vector<float> F(N);
float V = sqrt(U[1]*U[1]+U[2]*U[2]);
if (U[0]<=T1) {
M = (flag==0) ? (1-U[0]*Ka/T1) : exp(-U[0]*Ka/T1);
Fp = (flag==0) ? V1*Ka/T1/M : V1*Ka/T1; }
    else {
        M = 1;
        Fp = 0; }
    F[0] = 1;
    F[1] = -(B1+B2*V)*U[1]/M+Fp*U[1]/V;
    F[2] = -(B1+B2*V)*U[2]/M+Fp*U[2]/V-1;
    F[3] = U[1];
    F[4] = U[2];
    return F;
}

/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner)
    : TForm(Owner)
{}

/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)
{
try {
    frmUnit01->Caption = frmUnit01->Header0->Caption;
    flag = (RadioButton0->Checked) ? 0 : 1;
    Timer0->Enabled = false;
    dt = 0.01;

```

```

M0 = StrToFloat(Edit0->Text);
Kc = StrToFloat(Edit1->Text);
Mf = M0*(1+Kc);
Ka = (flag==0) ? Kc/(1+Kc) : log(1+Kc);
Yz = pi*StrToFloat(Edit5->Text)/180;
T0 = StrToFloat(Edit2->Text);
V0 = g*T0;
R0 = g*T0*T0/2;
B1 = StrToFloat(Edit7->Text)*T0/Mf;
B2 = StrToFloat(Edit8->Text)*2*R0/Mf;
V1 = StrToFloat(Edit6->Text)/V0;
dt = dt/T0;
T1 = 1;
X0 = StrToFloat(Edit3->Text)/R0;
Z0 = StrToFloat(Edit4->Text)/R0;

U0[0] = 0;
U0[1] = 0.001*V0*sin(Yz);
U0[2] = 0.001*V0*cos(Yz);
U0[3] = X0;
U0[4] = Z0;

    Button1Click((TObject *)Sender);
} catch(...) {
    ShowMessage("Помилка введення даних!");
}}
/*-----*/
void main(){
}
/*-----*/
void __fastcall TfrmUnit01::RadioButtonOClick(TObject *Sender)
{
    FormShow((TObject *)Sender);
}
/*-----*/

void __fastcall TfrmUnit01::ButtonOClick(TObject *Sender)
{
    Edit9->Text = FloatToStr(0);
    Edit10->Text = FloatToStr(0);
    Edit11->Text = FloatToStr(0);

```

```

Edit12->Text = FloatToStr(0);
Edit13->Text = FloatToStr(0);
Edit14->Text = FloatToStr(0);
FormShow((TObject *)Sender);
Button1Click((TObject *)Sender);
i = 0;
Timer0->Enabled = true;
Panel2->Caption = "Ракета в польоті";
}
/*-----*/
void __fastcall TfrmUnit01::Timer0Timer(TObject *Sender)
{
    fn_RK4exec0((TObject *)Sender);
    fn_RK4data0((TObject *)Sender);
    fn_RK4image0((TObject *)Sender);
    if (U[4] < 0) {
        Timer0->Enabled = false;
        Panel2->Caption = "Ракета зірвалася!!!";
    };
}
/*-----*/
void __fastcall TfrmUnit01::fn_RK4exec0(TObject *Sender)
{
    if (i == 0) { U = fn_RK4(fn_Force0, U0, dt); }
    Ux = U;
    U = fn_RK4(fn_Force0, U, dt);
    if (Ux[2]*U[2] < 0) { Uh = Ux;};
    i++;
}
/*-----*/
void __fastcall TfrmUnit01::fn_RK4data0(TObject *Sender) {
    float T, H;
    if (U[2] > 0) { T = 0; H = 0; }
    else { T = Uh[0]*T0; H = Uh[4]*R0; }
    Edit9->Text = FloatToStr(SimpleRoundTo(T,-2));
    Edit10->Text = FloatToStr(SimpleRoundTo(H,-2));
    Edit11->Text = FloatToStr(SimpleRoundTo(U[0]*T0,-2));
    Edit12->Text =
FloatToStr(SimpleRoundTo(sqrt(U[1]*U[1]+U[2]*U[2])*V0,-2));
    Edit13->Text =
FloatToStr(SimpleRoundTo((U[1]*U[1]+U[2]*U[2])*V0*V0/2/M0,-2));
}

```

```

        Edit14->Text = FloatToStr(SimpleRoundTo(U[3]*R0,-2));
    }
    /*-----*/
void __fastcall TfrmUnit01::fn_RK4image0(TObject *Sender) {
    dX = U[3]*R0*0.25/(1+Kc);
    dZ = -U[4]*R0*2.5/(1+Kc);
    if (i <= 1) {Image0->Canvas->Pen->Color=clWhite;}
    else
    if (U[0] <= T1) {Image0->Canvas->Pen->Color=clRed;}
    else
    {Image0->Canvas->Pen->Color=clBlue;}
    Image0->Canvas->LineTo(Cx+dX,Cz+dZ);
}
/*-----*/
void __fastcall TfrmUnit01::Button1Click(TObject *Sender)
{
    Cx=25;
    Cz=Image0->Height-25;
    Image0->Canvas->Pen->Width=5;
    Image0->Canvas->Pen->Color=clWhite;
    aRect = Rect(0,0,ClientWidth,ClientHeight);
    Image0->Canvas->FillRect(aRect);
    Image0->Canvas->FillRect(Rect(0,0,Image0->Width,Image0->Height));
    Image0->Canvas->MoveTo(Cx,Cz);
    Panel2->Caption = "Пакета на старті!!!";
}
/*-----*/
void __fastcall TfrmUnit01::Button2Click(TObject *Sender)
{
    frmUnit01->Close();
}
/*-----*/

```

В описовій частині коду тут використовується контейнерний тип даних, а саме 5-вимірний **vector**-функція узагальнених сил **vector<float> fn_Force0(vector<float> U)** аргументом якої є 5-вимірний **vector** узагальнених координат **vector<float> U**. У випадку руху ракети в площині $X, 0, Z$ до них відносяться t час, V_x, V_z — компоненти швидкості, R_x, R_z — компоненти координат ракети. У тілі функції **FormShow(TObject *Sender)** визначаються початкові умови для руху ракети, функція **Button0Click(TObject *Sender)** визначає початкову фазу руху ракети, функція **Timer0Timer(TObject *Sender)**

запускає ітерації методу Рунге-Кути. В тілі цієї функції використовується допоміжні функції

- `fn_RK4exec0((TObject *)Sender),`
- `fn_RK4data0((TObject *)Sender),`
- `fn_RK4image0((TObject *)Sender).`

Перша з них власне запускає ітераційний процес Рунге-Кути, друга на кожному кроці ітеративного процесу заповнює даними компоненти **Edit** на панелі "Динамічні показники руху ракети", а третя будує траєкторію польоту ракети на кожному кроці ітерації.

Перемикання режиму роботи двигуна відбувається за рахунок зміни стану компонентів `TRadioButton`, що викликає на виконання функцію `void __fastcall TfrmUnit01::RadioButon0Click(TObject *Sender).`

Перемикання режиму роботи додатку визначається компонентами `TButton`, "натискання" на які викликає на виконання асоційовані з ними функції.

2.3. Варіанти завдань до комп'ютерного практикуму

- Маса ракети без палива: [25; 30; 35; 40; 45;] кг
- Стартова маса ракети: [225; 250; 255; 260; 265;] кг
- Параметр швидкості згоряння палива: [0.25; 0.30; 0.35; 0.40; 0.45;] 1/сек
- Кут відхилення початкової швидкості ракети від вертикалі: [20; 5; 30; 35; 40;] град
- Відносна швидкість витікання гарячих газів: [1525; 1550; 1575; 1600; 1525;] м/сек

2.4. Підготовка до комп'ютерного практикуму

- Вивчити теоретичний матеріал щодо використання в практичному програмуванні контейнерних типів даних.
- Ознайомитися з теоретичними відомостями щодо теми комп'ютерного практикуму.
- Ознайомитися з рекомендаціями щодо виконання завдання

2.5. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.
- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project02.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project03.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям до виконання роботи, аналогічно Рис.2.1, розташуйте на головній формі проекту колекцію компонентів (2.3,) та надайте їм відповідне оформлення.
- У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <vector>`, `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "RK4.h"`.
- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).
- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Виконайте тестові розрахунки на прикладах з відомою відповіддю.

2.6. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості;

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочного файлу **RK4.h** та головного файлу **Unit01.cpp**;
 - висновки.

2.7. Контрольні запитання

- У чому заключаються переваги використання заголовочного файлу користувача **RK4.h**?
- Як означаються контейнерні типи даних `<vector>` та `<struct>`?
- У чому полягає головна відмінність між контейнерними типами даних `<vector>` та `<struct>`?
- У чому полягає відмінність контейнерних типів даних від динамічних масивів?
- У чому полягає різниця у доступ до елементів контейнерних типів даних `<vector>` та `<struct>`?
- Чи дозволяється використання контейнерного типу даних в якості типу даних для елементів іншого контейнерного типу даних?

3. Комп'ютерний практикум №3. Програмування додатку для побудови траєкторій польоту ракети за межі земної атмосфери з врахуванням стартових умов та режимів роботи двигуна

Мета роботи: Отримати поглиблений досвід використання контейнерних типів даних мови програмування C++ у практичному програмуванні.

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] візуальний додаток для побудови траєкторій польоту ракети за межі земної атмосфери з врахуванням стартових умов, режимів роботи двигуна, впливу атмосфери [7] з використанням контейнерних типів даних для наступних початкових умов: у момент $t = 0$ $\vec{r}(t = 0) = \vec{r}_0$, $\vec{v}(t = 0) = \vec{v}_0$, де $\vec{r} = (r_x, r_y, r_z)$, $\vec{v} = (v_x, v_y, v_z)$ — координати та швидкість руху ракети із змінною масою m .

Вхідні параметрами задачі:

- Координати стартового столу
- Маса ракети без палива
- Коефіцієнт Ціолковського
- Кут відхилення початкової швидкості ракети від вертикалі
- Час роботи двигуна ракети
- Відносна швидкість витікання гарячих газів
- Лінійний та квадратичний коефіцієнти тертя
- Барометричний показник

Розрахункові параметрами задачі:

- Швидкість ракети вздовж траєкторії польоту
- Віддалення супутника від планети
- Час перебування супутника в польоті
- Період обертання навколо планети

3.1. Теоретичні відомості

Політ ракети визначається 2-м законом Ньютона [7]

$$m \frac{d^2 \vec{r}}{dt^2} = m \vec{g} + \vec{f}_{fr} + \vec{f}_p \quad (3.1)$$

Тут g — напруженість гравітаційного поля Землі (Див.далі), $\vec{f}_{fr}$ — сила опору повітря, яка визначається формулою

$$\vec{f}_{fr} = -(k_1 + k_2 |\vec{v}|) \exp \left(-k_3 |\vec{r} - \vec{r}_0| \right) \vec{v}, \quad (3.2)$$

$$|\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}, \quad |\vec{r} - \vec{r}_0| = \sqrt{(r_x - r_{0x})^2 + (r_y - r_{0y})^2 + (r_z - r_{0z})^2},$$

де k_1, k_2 — коефіцієнти опору повітря 1-го та 2-го порядку, а коефіцієнт k_3 — описує зменшення опору повітря по мірі підйому ракети над поверхнею Землі, $\vec{f}_p = \vec{v}_p \frac{dm}{dt}$ — реактивна сила Мещерського [7]. Тут m — маса ракети на траєкторії польоту, а $\vec{v}_p$ — відносна швидкість витікання гарячих газів, яка вважається постійною за величиною.

На відміну від попередніх випадків тут ми враховуємо зменшення сили опору повітря при віддаленні від поверхні Землі за рахунок функції розподілу Больцмана [7] для концентрації газів в потенціальному полі.

Якщо координати центру Землі визначаються радіус-вектором $\vec{r}_0$, то вплив її гравітаційного поля на рух ракети можна описати формулою Ньютона [7] для сил гравітаційної взаємодії точкових мас

$$\vec{f}_g = m\vec{g}, \quad \vec{g} = -G \frac{M_3}{|\vec{r} - \vec{r}_0|^3}(\vec{r} - \vec{r}_0) = -g_0 \frac{R_3^2}{|\vec{r} - \vec{r}_0|^3}(\vec{r} - \vec{r}_0), \quad (3.3)$$

де $\vec{g}$ — напруженість гравітаційного поля Землі, а параметри M_3 та R_3 мають відповідно зміст гравітаційної маси Землі та її радіуса. Величина $\vec{g}_0$ — є прискорення вільного падіння.

Динамічні рівняння руху ракети в цьому випадку набувають вигляду [7]

$$m \frac{d^2 \vec{r}}{dt^2} = -g_0 \frac{R_3^2}{|\vec{r} - \vec{r}_0|^3}(\vec{r} - \vec{r}_0) - (k_1 + k_2|\vec{v}|) \exp\left(-k_3|\vec{r} - \vec{r}_0|\right) \vec{v} + \vec{v}_p \frac{dm}{dt}. \quad (3.4)$$

Якщо зв'язати систему відліку з центром Землі $\vec{r}_0 = 0$ то динамічні рівняння в безрозмірних змінних

$$\tau = \frac{t}{t_0}, \quad \vec{V} = \frac{\vec{v}}{v_0}, \quad \vec{R} = \frac{\vec{r}}{r_0}, \quad \mu = \frac{m}{m_f}$$

за умови $v_0 = g_0 t_0$, $r_0 = v_0 t_0$ набудуть вигляду

$$\begin{cases} \frac{d\vec{V}}{d\tau} = -\beta_4 \frac{\vec{R}}{R^3} - \frac{1}{\mu}(\beta_1 + \beta_2|\vec{V}|) \exp\left(-\beta_3 R\right) \vec{V} - \frac{\vec{V}}{|\vec{V}|} \frac{V_p}{\mu} \frac{d\mu}{d\tau} \\ \frac{d\vec{R}}{d\tau} = \vec{V} \end{cases} \quad (3.5)$$

де $\beta_1 = \frac{k_1 v_0}{m_f g_0}$, $\beta_2 = \frac{k_2 v_0^2}{m_f g_0}$, $\beta_3 = k_3 r_0$, $\beta_4 = \left(\frac{R_3}{r_0}\right)^2$, $V_p = \frac{v_p}{v_0}$, $\mu = \frac{m}{m_f}$ — безрозмірна маса ракети на твасекторії польоту, $m_f = m_0 + m_1$ де m_0 — власна маса ракети, m_1 — маса палива на старті при старті ракети. В якості ведучої константи виберемо величину t_0 , яка дорівнює часу роботи двигуна ракети.

Як і в попередніх проектах будемо розглядати два випадки режимів роботи двигуна ракети, а саме коли сила Мещерського є постійною величиною і коли прискорення руху ракеті є постійним.

В першому випадку маємо

$$m = m_f \mu(t) = m_f(1 - \alpha t) = m_f(1 - \alpha t) = m_f(1 - \alpha_1 \tau), \quad (3.6)$$

де $\alpha_1 = \alpha t_0$. Якщо врахувати, що час вигорання палива рівний як t_0 , то з

рівняння

$$\frac{m_0}{m_0 + m_1} = 1 - \alpha_1 \quad (3.7)$$

знайдемо параметр, що характеризує темп вигорання палива в процесі руху ракети $\alpha_1 = \frac{\chi}{1 + \chi}$, де $\chi = \frac{m_1}{m_0}$ — крефіцієнт Ціолковського. Сила Мещерського в цьому випадку очевидно набуде вигляду

$$\vec{F} = \alpha_1 \frac{V_p}{\mu} \frac{\vec{V}}{|\vec{V}|} \quad (3.8)$$

У другому випадку, коли ракета рухається з постійним прискоренням, темп витікання гарячих газів визначається формулою

$$m = m_f \mu(t) = m_f \exp(-\alpha t) = m_f \exp(-\alpha_1 \tau), \quad \alpha_1 = \alpha t_0. \quad (3.9)$$

В цьому випадку отримаємо $\alpha_1 = \ln(1 + \chi)$, а формула для реактивної сили тут набуває вигляду

$$\vec{F} = \alpha_1 V_p \frac{\vec{V}}{|\vec{V}|}. \quad (3.10)$$

Отримані рівня, що описують рух ракети за межі земної атмосфери, являються суттєво нелінійними. Тому їх розв'язки можна отримати лише методами обчислювальної математики.

При виконанні завдання потрібно розглянути два варіанти роботи двигуна ракети, а саме рух ракети під дією постійної сили Мещерського і рух ракети з постійним прискоренням. При моделюванні польоту ракети використати в якості моделі Землі коло в площині $0XZ$, а точку старту ракети вибрати в найвищій точці кола.

В програмній частині завдання є деякі моменти, які вже були розглянуті в комп'ютерних практикумах №1 та №2. В даній роботі також буде потрібно використати заголовочний файл користувача **RK4.h** для чисельного розв'язку задачі Коші (3.5), підключивши його за допомогою інструкції `#include "RK4.h"` до головного файлу проекту.

3.2. Рекомендації щодо виконання завдання

Приклад головної форми додатку приведено Рис.3.1 (див. далі). Там же приведені вхідні дані задачі, що задаються по замовчуванню. Для її побудови

необхідно використати такі компоненти

$$\begin{aligned}
 TForm &- 1 \text{ шт.}; & TPanel &- 2 \text{ шт.}; \\
 TRadioButton &- 2 \text{ шт.}; & TGroupBox &- 2 \text{ шт.}; \\
 TEdit &- 14 \text{ шт.}; & TLabel &- 14 \text{ шт.}; & TButton &- 3 \text{ шт.}; \\
 TImage &- 1 \text{ шт.}; & TTimer &- 1 \text{ шт.};
 \end{aligned}
 \tag{3.11}$$

які знаходяться на інструментальній панелі програмного середовища.

В головному файлі проекту знаходиться програмний код, який реалізує інтерфейс додатку (див. Рис.3.1) та його функціонал.

Приклад програмного коду головного файлу, що реалізує інтерфейс та функціонал додатку, можна написати у такому вигляді

```

/*- Unit01.cpp -----*/
#include <vcl.h>
#pragma hdrstop
#include <vector>
#include <Math.hpp>
#include <math.h>
using namespace std;
#include "Unit01.h"
#include "RK4.h"
/*-----*/
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmUnit01 *frmUnit01;
const float pi = 3.141563;
const float Rp = 6371*1e3;
const float V1 = 7910;
const float g = 9.8;
float H0 = 75000;
int flag,i,j,Cx,Cz,px,pz;
float M0,M,Mf,Kc,Ka,Yz,T0,V0,R0,B1,B2,B3,B4,
Tp,X0,Z0,Vp,Fp,V,R,dt,dX,dZ;
vector<float> U0(5),U(5),Ux(5),Uh(5);
TRect aRect;
vector<float> fn_Force0(vector<float> U) {
int N = U.size();
vector<float> F(N);
V = sqrt(U[1]*U[1]+U[2]*U[2]);
R = sqrt(U[3]*U[3]+U[4]*U[4]);
if (U[0]<=Tp) {

```

```

M = (flag==0) ? (1-U[0]*Ka) : exp(-U[0]*Ka);
Fp = (flag==0) ? Ka*Vp/M : Ka*Vp; }
else {
M = 1/(1+Kc);
Fp = 0; }
F[0] = 0;
F[1] = -(B1+B2*V)*exp(-B3*R)*U[1]/M+Fp*U[1]/V-B4*U[3]/R*R*R;
F[2] = -(B1+B2*V)*exp(-B3*R)*U[2]/M+Fp*U[2]/V-B4*U[4]/R*R*R;
F[3] = U[1];
F[4] = U[2];
return F;
}
/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner)
: TForm(Owner)
{}
/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)
{
try {
frmUnit01->Caption = frmUnit01->Header0->Caption;
flag = (RadioButton0->Checked) ? 0 : 1;
Timer0->Enabled = false;
M0 = StrToFloat(Edit0->Text)*1e7;;
Kc = StrToFloat(Edit1->Text);
Mf = M0*(1+Kc);
T0 = StrToFloat(Edit3->Text);
V0 = g*T0;
R0 = g*T0*T0;
Tp = 1;
Vp = StrToFloat(Edit2->Text)/V0;
Ka = (flag==0) ? Kc/(1+Kc)/Tp : log(1+Kc)/Tp;
B1 = StrToFloat(Edit4->Text)*T0/Mf;
B2 = StrToFloat(Edit5->Text)*R0/Mf;
B3 = V0*T0/StrToFloat(Edit6->Text);
B4 = (Rp/R0)*(Rp/R0);
Yz = pi*StrToFloat(Edit7->Text)/180;
dt = 1/T0;

X0 = 0;
Z0 = 0.1;//R0/R0;

```

```

U0[0] = 0;
U0[1] = 0.001*V0*sin(Yz);
U0[2] = 0.001*V0*cos(Yz);
U0[3] = X0;
U0[4] = Z0;

Cx = Image0->Width/2;
Cz = Image0->Height/2;

Button1Click((TObject *)Sender);
} catch(...) {
ShowMessage("Помилка введення даних!");
}}
/*-----*/
void main()
{}
/*-----*/
void __fastcall TfrmUnit01::RadioButton0Click(TObject *Sender)
{
FormShow((TObject *)Sender);
}
/*-----*/
void __fastcall TfrmUnit01::Button0Click(TObject *Sender)
{
Edit10->Text = FloatToStr(0);
Edit11->Text = FloatToStr(0);
Edit12->Text = FloatToStr(0);
Edit13->Text = FloatToStr(0);
Edit14->Text = FloatToStr(0);
Button1Click((TObject *)Sender);
FormShow((TObject *)Sender);
j = 0;
Timer0->Enabled = true;
Panel2->Caption = "Ракета в польоті";
}
/*-----*/
void __fastcall TfrmUnit01::Timer0Timer(TObject *Sender)
{
fn_RK4exec0((TObject *)Sender);
fn_RK4data0((TObject *)Sender);
fn_RK4image0((TObject *)Sender);
}

```

```

}
/*-----*/
void __fastcall TfrmUnit01::fn_RK4exec0(TObject *Sender)
{
if (j==0) { U = fn_RK4(fn_Force0, U0, dt); }
U = fn_RK4(fn_Force0, U, dt);
j++;
}
/*-----*/
void __fastcall TfrmUnit01::fn_RK4data0(TObject *Sender) {
    V = sqrt(U[1]*U[1]+U[2]*U[2]);
    R = sqrt(U[3]*U[3]+U[4]*U[4]);
Edit10->Text = FloatToStr((0));
Edit11->Text = FloatToStr((V*V0))*1e-3;
Edit12->Text = FloatToStr((R*R0))*1e-3;
Edit13->Text = FloatToStr((U[0]*T0))/3600;
Edit14->Text = FloatToStr((0));
}
/*-----*/
void __fastcall TfrmUnit01::fn_RK4image0(TObject *Sender) {
dX = U[3]*Rp*1e-5;
dZ = -U[4]*Rp*1e-5;
if (j <= 1) {Image0->Canvas->Pen->Color=clWhite;}
else
if (U[0] <= Tp) {
if (j<=5) {
Image0->Canvas->Pen->Width=15;
Image0->Canvas->Pen->Color=clGreen;
}
else {
Image0->Canvas->Pen->Width=5;
if(j<=25) {Image0->Canvas->Pen->Color=clGreen;}
else {Image0->Canvas->Pen->Color=clRed; }
}
}
else {Image0->Canvas->Pen->Color=clBlue;}
Image0->Canvas->LineTo(Cx+dX,Cz+dZ);
}
/*-----*/
void __fastcall TfrmUnit01::Button1Click(TObject *Sender)
{

```

```

Timer0->Enabled = false;
Image0->Canvas->Pen->Width=5;
Image0->Canvas->Pen->Color=clWhite;
aRect = Rect(0,0,ClientWidth,ClientHeight);
Image0->Canvas->FillRect(aRect);
Image0->Canvas->FillRect(Rect(0,0,Image0->Width,Image0->Height));
Image0->Canvas->MoveTo(Cx,Cz);
Panel2->Caption = "Пакета на старті!!!";
}
/*-----*/
void __fastcall TfrmUnit01::Button2Click(TObject *Sender)
{
frmUnit01->Close();
}
/*-----*/

```

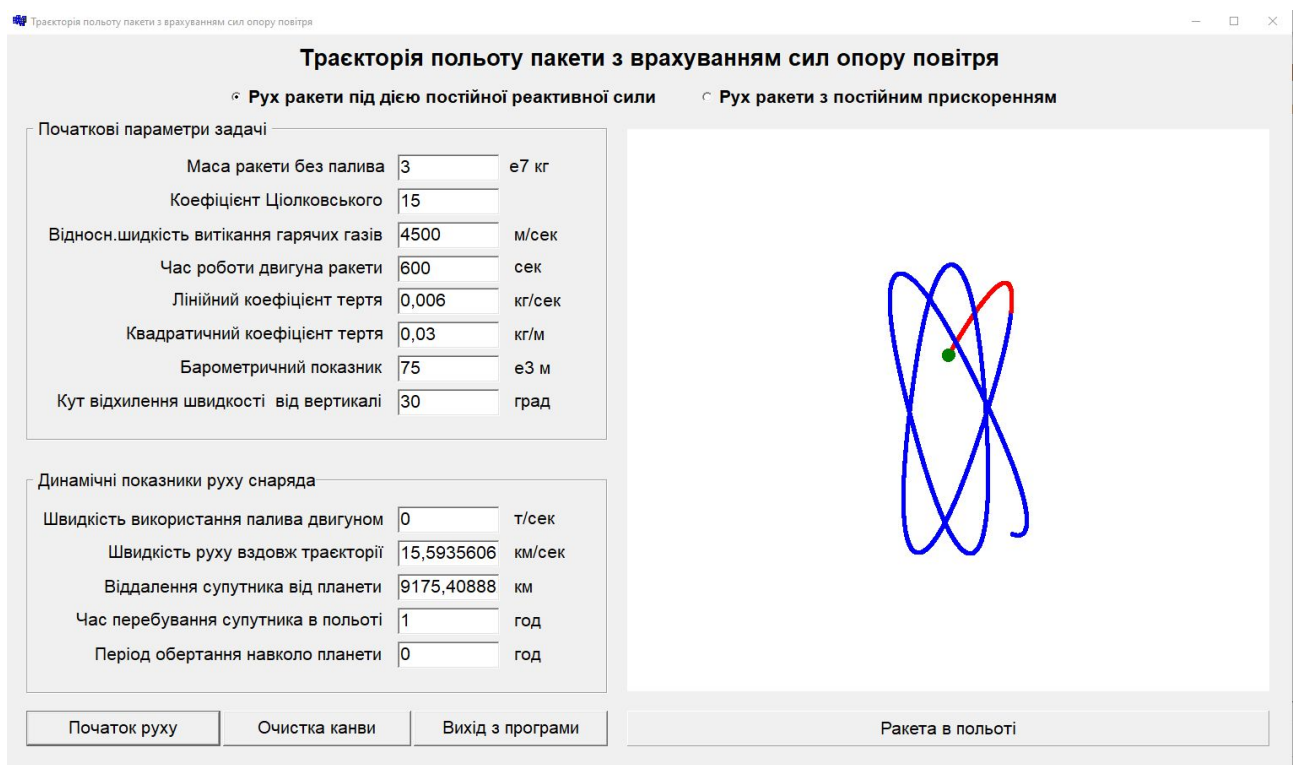


Рис. 3.1. Візуальний додаток для розрахунку траєкторії руху ракети

В описовій частині коду тут використовується контейнерний тип даних, а саме 5-вимірна vector-функція узагальнених сил **vector<float> fn_Force0(vector<float> U)** аргументом якої є 5-вимірний vector узагальнених координат **vector<float> U**. У випадку руху ракети в площині $X, 0, Z$ до них відносяться t час, V_x, V_z — компоненти швидкості, R_x, R_z — компоненти координат ракети. У тілі функції **FormShow(TObject *Sender)** визначаються початкові умови для руху ракети, функція **Button0Click(TObject *Sender)** визначає початкову фазу руху ракети, функція **Timer0Timer(TObject *Sender)**

запускає ітерації методу Рунге-Кути. В тілі цієї функції використовується допоміжні функції

- `fn_RK4exec0((TObject *)Sender),`
- `fn_RK4data0((TObject *)Sender),`
- `fn_RK4image0((TObject *)Sender).`

Перша з них власне запускає ітераційний процес Рунге-Кути, друга на кожному кроці ітеративного процесу заповнює даними компоненти **Edit** на панелі "Динамічні показники руху ракети", а третя будує траєкторію польоту ракети на кожному кроці ітерації.

Перемикання режиму роботи двигуна відбувається за рахунок зміни стану компонентів `TRadioButton`, що викликає на виконання функцію `void __fastcall TfrmUnit01::RadioButton0Click(TObject *Sender).`

Перемикання режиму роботи додатку визначається компонентами `TButton`, "натискання" на які викликає на виконання асоційовані з ними функції.

3.3. Варіанти завдань до комп'ютерного практикуму

- Координати стартового столу: $[0,175; 0,180; 0,185; 0,190; 0,195;]$ м
- Маса ракети без палива: $[3; 3.25; 3.5; 3.75; 4;]$ 10^7 кг
- Коефіцієнт Ціолковського: $[15; 15.25; 15.5; 15.75; 16;]$
- Кут відхилення початкової швидкості ракети від вертикалі: $[20; 5; 30; 35; 40;]$ град
- Час роботи двигуна ракети: $[600; 625; 650; 675; 700;]$ сек
- Відносна швидкість витікання гарячих газів: $[4500; 4525; 4550; 4575; 4600;]$ м/сек
- Лінійний та квадратичний коефіцієнти тертя: $[0.0075, 0.0343; 0.0067, 0.0345; 0.0085, 0.0242; 0.0066, 0.0442; 0.0035, 0.0542;]$
- Барометричний показник: $[75; 75.25; 75.5; 75.75; 76;]$ 10^3 м

3.4. Підготовка до комп'ютерного практикуму

- Розглянути теоретичний матеріал щодо використання контейнерних типів даних для розв'язку задачі Коші методом Рунге-Кути 4-го порядку точності.

- Ознайомитися з теоретичними відомостями щодо теми комп'ютерного практикуму.
- Ознайомитися з рекомендаціями щодо виконання завдання.

3.5. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.
- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project03.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project04.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям щодо виконання роботи, аналогічно Рис.3.1, розташуйте на головній формі проекту колекцію компонентів (5.1,) та надайте їх відповідне оформлення.
- У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <vector>`, `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "RK4.h"`.
- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).
- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Виконайте тестові розрахунки на прикладах з відомою відповіддю.

3.6. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості.

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочного файлу **RK4.h** та головного файлу **Unit01.cpp.**;
 - висновки.

3.7. Контрольні запитання

- У чому заключаються переваги використання заголовочного файлу користувача **RK4.h**?
- У чому полягає відмінність контейнерних типів даних від динамічних масивів?
- Як означаються контейнерні типи даних `<vector>` та `<struct>`?
- У чому полягає головна відмінність між контейнерними типами даних `<vector>` та `<struct>`?
- У чому полягає різниця у доступ до елементів контейнерних типів даних `<vector>` та `<struct>`?
- Чи дозволяється використання контейнерного типу даних в якості типу даних для елементів іншого контейнерного типу даних?

4. Комп'ютерний практикум №4. Програмування додатку "Калькулятор комплексної алгебри" на основі контейнерного типу даних "struct"

Мета роботи: Отримати досвід використання контейнерного типу даних `<struct>` та технології перезавантаження операцій мови програмування C++ у практичному програмуванні [2].

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] візуальний додаток для основних операцій алгебри комплексних чисел

- додавання, віднімання, множення, ділення;
- комплексного спряження;
- обчислення модуля та фази;
- піднесення до довільного степеня

з використання контейнерного типу даних `<struct>` та технології перезавантаження операцій.

При реалізації додатку передбачити програмування комірки пам'яті та функції очищення полів.

4.1. Теоретичні відомості

При розробці додатку потрібно скористатись відомими формулами теорії функцій комплексної змінної $z = x + iy$

- $z = z_1 + z_2 = (x_1 + x_2) + i(y_1 + y_2)$ — складання;
- $z = z_1 - z_2 = (x_1 - x_2) + i(y_1 - y_2)$ — віднімання;
- $z = z_1 \cdot z_2 = (x_1 \cdot x_2 - y_1 \cdot y_2) + i(x_1 \cdot y_2 + x_2 \cdot y_1)$ — множення;
- $z = z_1 / z_2 = \frac{(x_1 \cdot x_2 + y_1 \cdot y_2) + i(x_2 \cdot y_1 - x_1 \cdot y_2)}{x_2^2 + y_2^2}$ — ділення;
- $z^* = x - iy$ — спряження комплексно числа;
- $|z| = \sqrt{x^2 + y^2}$ — модуль комплексно числа;
- $\varphi = \arctg\left(\frac{y}{x}\right)$ — фаза комплексно числа;
- $z^n = |z|^n (\cos(n \cdot \varphi) + i \sin(n \cdot \varphi))$ — піднесення до n -го степеня,

які визначають основні операції алгебри комплексних чисел $z = x + iy$.

При програмуванні додатку потрібно використати контейнерний тип даних `<struct>`

```
struct Complex{
public:
double ReZ;
double ImZ;};
```

та перезавантаження операцій, програмування яких доцільно винести в окремий універсальний заголовочний файл **Complex.h**

```

/*- Complex.h -----*/
#ifndef ComplexH
#define ComplexH

using namespace std;
/*-----*/
struct Complex{
public:
double ReZ;
double ImZ;
};

//Операторна функція для складання двох об'єктів:
Complex operator+(Complex U, Complex V){
Complex Z;
Z.ReZ=U.ReZ+V.ReZ;
Z.ImZ=U.ImZ+V.ImZ;
return Z;}

//Операторна функція для віднімання двох об'єктів:
Complex operator-(Complex U, Complex V){
Complex Z;
Z.ReZ=U.ReZ-V.ReZ;
Z.ImZ=U.ImZ-V.ImZ;
return Z;}

//Операторна функція для складання числа і об'єкту:
Complex operator+(double U, Complex V){
Complex Z;
Z.ReZ=U+V.ReZ;
Z.ImZ=V.ImZ;
return Z;}

//Операторна функція для складання об'єкту і числа:
Complex operator+(Complex V, double U){
Complex Z;
Z.ReZ=U+V.ReZ;
Z.ImZ=V.ImZ;
return Z;}

//Операторна функція для віднімання із числа об'єкта:
Complex operator-(double U, Complex V){
Complex Z;

```

```

Z.ReZ=U-V.ReZ;
Z.ImZ=-V.ImZ;
return Z;}
//Операторна функція для віднімання із об'єкту числа:
Complex operator-(Complex V, double U){
Complex Z;
Z.ReZ=V.ReZ-U;
Z.ImZ=V.ImZ;
return Z;}
//Операторна функція для перемноження об'єктів:
Complex operator*(Complex U, Complex V){
Complex Z;
Z.ReZ=U.ReZ*V.ReZ-U.ImZ*V.ImZ;
Z.ImZ=U.ReZ*V.ImZ+U.ImZ*V.ReZ;
return Z;}
//Операторна функція для ділення об'єктів:
Complex operator/(Complex U, Complex V){
Complex Z;
if (V.ReZ*V.ReZ+V.ImZ*V.ImZ > 0) {
Z.ReZ=(U.ReZ*V.ReZ+U.ImZ*V.ImZ)/(V.ReZ*V.ReZ+V.ImZ*V.ImZ);
Z.ImZ=(U.ImZ*V.ReZ-U.ReZ*V.ImZ)/(V.ReZ*V.ReZ+V.ImZ*V.ImZ); }
else {Z.ReZ=0; Z.ImZ=0;}
return Z;}
/*-----*/
#endif

```

який можна буде використовувати і в майбутніх проектах.

Ключовою властивістю контейнерного типу даних **struct** є те, що їх можна використовувати (наряду із звичайними змінними) в якості аргументів функцій та їх значень.

4.2. Рекомендації щодо виконання завдання

Приклад головної форми додатку приведено Рис.4.1. Там же приведені вхідні дані задачі, що задаються по замовчуванню. Для її побудови необхідно використати такі компоненти

$$\begin{aligned}
 TForm &- 1 \text{ шт.}; & TGroupBox &- 1 \text{ шт.}; \\
 TEdit &- 4 \text{ шт.}; & TLabel &- 4 \text{ шт.}; & TButton &- 12 \text{ шт.},
 \end{aligned}
 \tag{4.1}$$

які знаходяться на інструментальній панелі.

В головному файлі проекту знаходиться програмний код, який реалізує

інтерфейс додатку (див. Рис.4.1) та його функціонал.

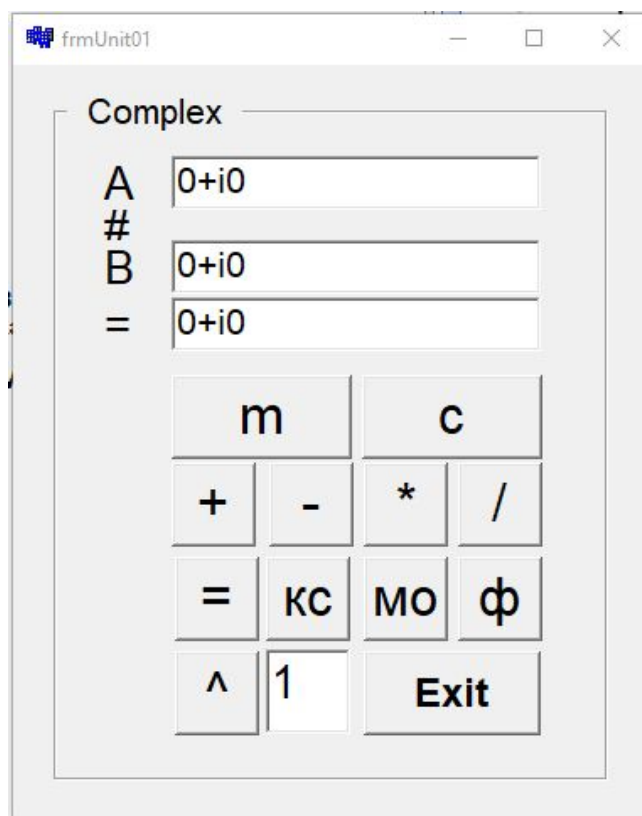


Рис. 4.1. "Калькулятор комплексної алгебри"

Приклад програмного коду головного файлу, що реалізує інтерфейс та функціонал додатку, можна написати у такому вигляді

```

/*- Unit01.cpp -----*/
#include <string>
#include <iostream>
#include <vcl.h>
#include <Math.hpp>
#include <math.h>
#pragma hdrstop

using namespace std;

#include "Complex.h"
#include "Unit01.h"
/*-----*/
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmUnit01 *frmUnit01;
TEdit *Edit_T;
Complex A,B,C,T;

```

```

AnsiString X,Y,flag;
int j,k,sign,EditNo;

Complex fn_CNumber(TEdit *Edit0) {
    Complex Q;
    AnsiString X = Edit0->Text;
    int j = X.AnsiPos("i");
    if (j==0) {
        Q.ReZ = StrToFloat(X);
        Q.ImZ = 0;
    } else
    {
        sign = (X.AnsiPos("+") > 0) ? 1 : -1;
        Q.ReZ = StrToFloat(X.SubString(0,j-2));
        Q.ImZ = sign*StrToFloat(X.SubString(j+1,50));
    }
    return Q;
}

/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner):
TForm(Owner)
{
}
/*-----*/
void main()
{}
/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)
{
    Edit0->Text = "0+i0";
    Edit1->Text = "0+i0";
    Edit2->Text = "0+i0";
}
/*-----*/
void __fastcall TfrmUnit01::Edit0Enter(TObject *Sender)
{
    Edit_T = ((TEdit*)Sender);
    T = fn_CNumber(Edit_T);
}
/*-----*/
void __fastcall TfrmUnit01::Edit0KeyUp(TObject *Sender,
WORD &Key, TShiftState Shift)

```

```

{
    Edit_T = ((TEdit*)Sender);
    AnsiString Txt = Edit_T->Text;
    int j = Txt.AnsiPos("+")+Txt.AnsiPos("-");
    int k = Txt.AnsiPos("i");
    if ((j>0)&&(k==0)) {Edit_T->Text = Edit_T->Text+"i";}
    Edit_T->SelStart=Edit_T->Text.Length();
}
/*-----*/
void __fastcall TfrmUnit01::Button0Click(TObject *Sender)
{
    Label0->Caption = ((TButton *)Sender)->Caption;
}
/*-----*/
void __fastcall TfrmUnit01::Button4Click(TObject *Sender)
{
    T = fn_CNumber(Edit_T);
        T.ReZ = T.ReZ;
        T.ImZ = (-1)*T.ImZ;
    flag = (T.ImZ<0) ? "-" : "+";
    Edit_T->Text = FloatToStr(T.ReZ)+flag+"i"+
    FloatToStr(fabs(T.ImZ))+"";
}
/*-----*/
void __fastcall TfrmUnit01::Button5Click(TObject *Sender)
{
    T = fn_CNumber(Edit_T);
    double M;
    M=SimpleRoundTo(sqrt(T.ReZ*T.ReZ+T.ImZ*T.ImZ),-6);
    Edit_T->Text = FloatToStr(M);
}
/*-----*/
void __fastcall TfrmUnit01::Button6Click(TObject *Sender)
{
    T = fn_CNumber(Edit_T);
    double Y;
    if (T.ReZ!=0) {Y = SimpleRoundTo(atan1(T.ImZ/T.ReZ),-6);}
    else { Y = 0; }
    Edit_T->Text = FloatToStr(Y);
}
/*-----*/

```

```

void __fastcall TfrmUnit01::Button7Click(TObject *Sender)
{
    T = fn_CNumber(Edit_T);
    double M,Y;
    int N = StrToInt(Edit3->Text);
    int j = Edit_T->Text.AnsiPos("i");
    if (j==0) {T.ImZ=0;}
    M=sqrt( T.ReZ*T.ReZ+T.ImZ*T.ImZ );
    M = pow(M,N);
    if (T.ReZ!=0) { Y = atan1(T.ImZ/T.ReZ); }
    else { Y = 0; }
    T.ReZ = SimpleRoundTo(M*cos(N*Y),-6);
    T.ImZ = SimpleRoundTo(M*sin(N*Y),-6);
    flag = (T.ImZ<0) ? "-" : "+";
    Edit_T->Text = FloatToStr(T.ReZ)+flag+"i"+
    FloatToStr(fabs(T.ImZ))+"";
}
/*-----*/
void __fastcall TfrmUnit01::Button9Click(TObject *Sender)
{
    FormShow((TObject *)Sender);
}
/*-----*/
void __fastcall TfrmUnit01::Button10Click(TObject *Sender)
{
    A = fn_CNumber(Edit0);
    B = fn_CNumber(Edit1);
    if (Label0->Caption=="+") {C=A+B;} else
    if (Label0->Caption=="-") {C=A-B;} else
    if (Label0->Caption=="*") {C=A*B;} else
    if (Label0->Caption=="/") {C=A/B;}
    flag = (C.ImZ<0) ? "-" : "+";
    Edit2->Text = FloatToStr(C.ReZ)+flag+"i"+
    FloatToStr(fabs(C.ImZ))+"";
}
/*-----*/
void __fastcall TfrmUnit01::Button11Click(TObject *Sender)
{
    frmUnit01->Close();
}
/*-----*/

```

Зверніть увагу на те, що використання контейнерних типів даних `<struct>` та перезавантаження операцій, які підключаються до головного файлу проекту інструкцією `#include "Complex.h` звільняє програміста від зайвого програмування алгебраїчних операцій над комплексними числами.

4.3. Підготовка до комп'ютерного практикуму

- Вивчити теоретичний матеріал щодо контейнерних типів даних `<struct>` та перезавантаження операцій.
- Ознайомитися з теоретичними відомостями щодо теми комп'ютерного практикуму.
- Ознайомитися з рекомендаціями щодо виконання завдання.

4.4. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.
- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project04.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project02.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям щодо виконання роботи, аналогічно Рис.4.1, розташуйте на головній формі проекту колекцію компонентів 4.1, та надайте їх відповідне оформлення.
- В папці з файлами проекту створіть заголовочний файл **Complex.h** та наповніть його кодом, який реалізує алгебру комплексних чисел (див. Рекомендації щодо ...). У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "Complex.h"`.

- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).
- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Виконайте тестові розрахунки на прикладах з відомою відповіддю.

4.5. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості.

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочного файлу **Complex.h** та головного файлу **Unit01.cpp**;
 - висновки.

4.6. Контрольні запитання

- В чому заключаються переваги використання заголовочного файлу користувача **Complex.h**?
- Як означається контейнерний тип даних `<struct>`?
- Чи можуть зберігатися в контейнерні типу `<struct>` елементи, що відносяться до різних типів даних?
- У чому полягає відмінність контейнерного типу даних `<struct>` від динамічних масивів?
- Як здійснюється доступ до елементів контейнерного типу даних `<struct>`?
- Чи дозволяється використання контейнерного типу даних `<struct>` в якості типу даних для елементів контейнерного типу даних `<struct>`?

5. Комп'ютерний практикум №5. Програмування додатку для файлової бази даних на основі вкладених структур (тип даних "struct")

Мета роботи: Отримати досвід використання вкладених структур для програмування файлової бази даних.

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] додаток для файлової бази даних на основі вкладених структур для контролю за самостійною роботою студентів з навчальних дисциплін.

Вхідні параметрами задачі:

- Назва академ групи;
- Кількість студентів в академгрупі;
- Назва навальної дисципліні;
- Список студентів академгрупи;
- Результати оцінювання виконання індивідуальних завдань.

Розрахункові параметрами задачі:

- Сумарні бали за семестр.

5.1. Теоретичні відомості

В процесі виконання завдання комп'ютерного практикуму потрібно створити додаток для файлової бази даних до якої передбачається заносити перелік навчальних дисциплін, академгруп, кількості студентів в академгрупах, склад академгруп, кількість балів за виконання кожного індивідуального завдання. В якості розрахункового параметра виступає поточний сумарний бал за виконання індивідуальних завдань (див. Рис.5.1).

В основі проекту файлової бази даних ланцюжок зв'язаних структур [2], коди яких локалізуються у заголовочному файлі *Students.h*

```
#ifndef StudentsH
#define StudentsH
/*-----*/
struct Sbjs {
char sbj[150];
```

```

};
/*-----*/
struct Agrps {
Sbjs Sbjs0;
char agrp[10];
int Studs;
};
/*-----*/
struct Strms {
Agrps Agrps0;
char name[30], surname[20];
};
/*-----*/
struct Tests {
Strms Studs0;
int Data01, Data02, Data03,
Data04, Data05, Data06, Data07, Data08;
int SumData(){
int Res0=Data01+Data02+Data03+
Data04+Data05+Data06+Data07+Data08;
return Res0;};
};
/*-----*/
#endif

```

Цей файл звичайним чином підключається до головного файлу проекту Unit01.cpp.

Контроль успішності самостійної роботи студентів

Склад академгрупи: кафедри СКЛА: студентів

Назва навчальної дисципліни:

№	Прізвище	Ім'я	ІндЗавд. №1	ІндЗавд. №2	ІндЗавд. №3	ІндЗавд. №4	ІндЗавд. №5	ІндЗавд. №6	ІндЗавд. №7	ІндЗавд. №8	Сума балів
1	Бровченко	Роман	7	5	8	6					26
2	Васьков	Володимир	9	7	8	9	10				43
3	Герасименко	Іван	6	8	7	6	9				36
4	Романенко	Олег	9	8	6	8	9				40
5	Хромов	Арсен	5	6	7	8					26

Buttons: To file, From file, Close

Рис. 5.1. Інтерфейс файлової бази даних

5.2. Рекомендації щодо виконання завдання

Приклад головної форми додатку приведено на Рис.5.1. Для її побудови необхідно використати такі компоненти

$$\begin{aligned}
 TForm &- 1 \text{ шт.}; & TPanel &- 4 \text{ шт.}; \\
 TComboBox &- 2 \text{ шт.}; & TGroupBox &- 1 \text{ шт.}; \\
 TEdit &- 2 \text{ шт.}; & TLabel &- 3 \text{ шт.}; \\
 TStringGrid &- 1 \text{ шт.}; & TButton &- 3 \text{ шт.};
 \end{aligned}
 \tag{5.1}$$

які знаходяться на інструментальній панелі програмного середовища.

Сховище бази даних оформляється у вигляді набору файлів. Їх імена формуються таким чином: *FileIJ.dat*, де *I* — індекс навчальної дисципліни, а *J* — індекс академгрупи.

В головному файлі пректу знаходиться програмний код, який реалізує інтерфейс додатку (див. Рис.5.1) та його функціонал. Його можна написати у такому вигляді

```

/*- Unit01.cpp -----*/
#include <vcl.h>
#pragma hdrstop
#include <fstream.h>
#include "Unit01.h"
#include "Students.h"
/*-----*/
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmUnit01 *frmUnit01;
AnsiString fn;
fstream f;
char *filename=new char[30];
int SumData;
Tests clsTst;
/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner)
    : TForm(Owner)
{
}
/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)
{
    frmUnit01->Width = 1850;

```

```

frmUnit01->Height = 900;
Button0->Width = 125;
Button1->Width = Button0->Width;
Button2->Width = Button0->Width;
Button2->Left = Panel1->Width-Button2->Width-15;
Button1->Left = Button2->Left-Button1->Width-1;
Button0->Left = Button1->Left-Button0->Width-1;
StringGrid0->Cells[0][0] = " №";
StringGrid0->Cells[1][0] = " Прізвище";
StringGrid0->Cells[2][0] = " Ім'я";
StringGrid0->Cells[11][0] = " Сума балів";
for (int i=1; i<StringGrid0->RowCount; i++) {
StringGrid0->Cells[0][i] = " "+IntToStr(i);}
for (int j=3; j<StringGrid0->ColCount-1; j++) {
StringGrid0->Cells[j][0] = " ІндЗавд.№"+IntToStr(j-2);}
Edit0Change((TObject *)Sender);
ComboBox1Click((TObject *)Sender);
f.clear();
f.close();
if(f) {Button1Click((TObject *)Sender);}
}
/*-----*/
void __fastcall TfrmUnit01::Edit0Change(TObject *Sender)
{
StringGrid0->RowCount = StrToInt(Edit0->Text)+1;
for (int i=1; i<StringGrid0->RowCount+1; i++) {
StringGrid0->Cells[0][i] = " "+IntToStr(i);}
}
/*-----*/
void __fastcall TfrmUnit01::Button0Click(TObject *Sender)
{
remove(filename);
f.open(filename,ios::out|ios::app|ios::binary);
f.clear();
if(!f) {ShowMessage("Помилка відкриття файлу"); return;}
for (int i=1; i < StringGrid0->RowCount; i++) {
f.seekp(0,ios::end);

strcpy(clsTst.Studs0.Agrps0.Sbjs0.sbj,  ComboBox1->Text.c_str());
strcpy(clsTst.Studs0.Agrps0.agrp,  ComboBox0->Text.c_str());
clsTst.Studs0.Agrps0.Studs = StrToInt(Edit0->Text);
strcpy(clsTst.Studs0.name, StringGrid0->Cells[1][i].c_str());

```

```

strcpy(clsTst.Studs0.surname, StringGrid0->Cells[2][i].c_str());
clsTst.Data01 = (StringGrid0->Cells[3][i]=="")?0:
StrToInt(StringGrid0->Cells[3][i]);
clsTst.Data02 = (StringGrid0->Cells[4][i]=="")?0:
StrToInt(StringGrid0->Cells[4][i]);
clsTst.Data03 = (StringGrid0->Cells[5][i]=="")?0:
StrToInt(StringGrid0->Cells[5][i]);
clsTst.Data04 = (StringGrid0->Cells[6][i]=="")?0:
StrToInt(StringGrid0->Cells[6][i]);
clsTst.Data05 = (StringGrid0->Cells[7][i]=="")?0:
StrToInt(StringGrid0->Cells[7][i]);
clsTst.Data06 = (StringGrid0->Cells[8][i]=="")?0:
StrToInt(StringGrid0->Cells[8][i]);
clsTst.Data07 = (StringGrid0->Cells[9][i]=="")?0:
StrToInt(StringGrid0->Cells[9][i]);
clsTst.Data08 = (StringGrid0->Cells[10][i]=="")?0:
StrToInt(StringGrid0->Cells[10][i]);
SumData = clsTst.SumData();
StringGrid0->Cells[11][i] = IntToStr(SumData);

f.write((char*)&clsTst, sizeof(clsTst));}
f.close();
}
/*-----*/
void __fastcall TfrmUnit01::Button1Click(TObject *Sender)
{
if(f) {f.close();};
f.open(filename,ios::out|ios::in|ios::binary);
f.clear();
if(!f)
{ShowMessage("Помилка відкриття файла"); return;}
int i = 1;
while(f.read((char*)&clsTst, sizeof(clsTst))) {
Edit0->Text = IntToStr(clsTst.Studs0.Agrps0.Studs);
ComboBox0->Text = AnsiString(clsTst.Studs0.Agrps0.agrp);
ComboBox1->Text = AnsiString(clsTst.Studs0.Agrps0.Sbjs0.sbj);
StringGrid0->Cells[1][i] = AnsiString(clsTst.Studs0.name);
StringGrid0->Cells[2][i] = AnsiString(clsTst.Studs0.surname);
StringGrid0->Cells[3][i] = (clsTst.Data01==0)?AnsiString(""):
IntToStr(clsTst.Data01);
StringGrid0->Cells[4][i] = (clsTst.Data02==0)?AnsiString(""):

```

```

IntToStr(clsTst.Data02);
StringGrid0->Cells[5][i] = (clsTst.Data03==0)?AnsiString(""):
IntToStr(clsTst.Data03);
StringGrid0->Cells[6][i] = (clsTst.Data04==0)?AnsiString(""):
IntToStr(clsTst.Data04);
StringGrid0->Cells[7][i] = (clsTst.Data05==0)?AnsiString(""):
IntToStr(clsTst.Data05);
StringGrid0->Cells[8][i] = (clsTst.Data06==0)?AnsiString(""):
IntToStr(clsTst.Data06);
StringGrid0->Cells[9][i] = (clsTst.Data07==0)?AnsiString(""):
IntToStr(clsTst.Data07);
StringGrid0->Cells[10][i] = (clsTst.Data08==0)?AnsiString(""):
IntToStr(clsTst.Data08);
Edit0Change((TObject *)Sender);
SumData = clsTst.SumData();
StringGrid0->Cells[11][i] = IntToStr(SumData);
i++;
}
f.close();
}
/*-----*/
void __fastcall TfrmUnit01::ComboBox1Click(TObject *Sender)
{
fn = "File"+IntToStr(ComboBox1->ItemIndex)+
IntToStr(ComboBox0->ItemIndex)+".dat";
strcpy(filename,fn.c_str());
Button1Click((TObject *)Sender);
}
/*-----*/
void __fastcall TfrmUnit01::Button2Click(TObject *Sender)
{
frmUnit01->Close();
}
/*-----*/

```

5.3. Завдання до комп'ютерного практикуму

- Занести до БД назву своєї академ.групи;
- Занести у відповідний компонент інтерфейсу кількість студентів в академ.групі;

- Занести до БД не менше 5 навчальних дисциплін з робочої навчальної програми;
- Занести до БД список студентів академгрупи;
- Занести до БД результати оцінювання виконання модульних контрольних робіт

5.4. Підготовка до комп'ютерного практикуму

- Розглянути теоретичний матеріал щодо контейнерного типу даних *struct*, а також способів формування вкладених структур із типу даних *struct*.
- Розглянути теоретичний матеріал щодо типу даних *file* та їх маніпуляціями.
- Ознайомитися з теоретичними відомостями щодо теми комп'ютерного практикуму.
- Ознайомитися з рекомендаціями щодо виконання завдання.

5.5. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.
- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project05.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project04.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям щодо виконання роботи, аналогічно Рис.5.1, розташуйте на головній формі проекту колекцію компонентів (5.1) та надайте їх відповідне оформлення.

- У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "Students.h"`.
- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).
- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Запустіть додаток. На головній формі виберіть у випадajoчих списках назву вашої академгруппи та одну із навчальних дисциплін. У відповідному компоненті вкажіть кількість студентів у вашій академгруппі. У сітці заведіть склад академгруппи та виставлені викладачем бали за виконання студентами індивідуальних завдань. Натисніть кнопку **To file** для занесення даних до файлу бази даних. Закрийте додаток натиснувши кнопку **Close**. Повторно запустіть додаток. Переконайтесь, що в компонентах головної форми коректно візуалізувалися раніше занесені дані. Закрийте додаток.

5.6. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості.

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочного файлу **Students.h** та головного файлу **Unit01.cpp**;
 - аналіз роботи додатку та висновки щодо його практичного застосування.

5.7. Контрольні запитання

- Яким чином визначаються окремі та вкладені структури?
- Яким чином об'являється тип даних *file*?
- Яким чином програмується запис та зчитування даних з файлу?
- Чому при побудові файлової бази даних доцільно використовувати вкладені структури?
- Як Ви розумієте зміст таких термінів, як властимості та методи структури?

6. Комп'ютерний практикум №6. Програмування класу для розв'язку задачі Коші для довільної системи диференціальних рівнянь 1-го порядку у випадку немонотонних динамічних систем

Мета роботи: Отримати досвід побудови класів з використанням контейнерних типів даних мови програмування C++ у практичному програмуванні.

Запрограмувати перезавантаження операцій для контейнерних типів даних **vector<double>** **dvector** та **vector<vector<double> >** **dmatrix**.

Запрограмувати клас **ClsDES4** для розв'язку задачі Коші довільної системи диференціальних рівнянь 1-го порядку методом Адамса-Башфорта 4-порядку [2].

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] додаток для тестування класу для розв'язків задачі Коші у випадку немонотонних динамічних систем методом Адамса-Башфорта 4-порядку [2] на прикладі динаміки гармонічного осцилятора з дисипативним елементом під дією зовнішньої сили. Передбачити збереження результатів розрахунку у табличному форматі та візуалізації табличних даних у вигляді графіку.

Вхідні параметри задачі:

- Параметр затухання коливань
- Амплітуда коливань зовнішньої сили
- Частота коливань зовнішньої сили

- Початкове значення амплітуда коливань осцилятора
- Початкове значення швидкості коливань осцилятора
- Початок коливального руху
- Закінчення коливального руху
- Крок ітераційного процесу

Розрахункові параметрами задачі:

- Амплітуда зміщень вимушених коливань осцилятора
- Амплітуда швидкості вимушених коливань осцилятора
- Резонансна частота осцилятора

6.1. Теоретичні відомості

Оскільки тип даних **vector** відноситься до вбудованого класу мови програмування C++, то перезантаження операцій для введених нами типів даних **vector<double>** **dvector** та **vector<vector<double> >** **dmatrix** програмується стандартним чином. Але для зручності їх використання виконаємо для них переозначення

```
typedef vector<double> dvector;
typedef vector<vector<double> > dmatrix;
```

Тоді заголовочний файл **Matrix0.h** з програмою перезавантаження операцій для нових типів даних **dvector** та **dmatrix** матиме таке наповнення

```
// Matrix0.h -----
#ifndef Matrix0H
#define Matrix0H
#include <vector>
using namespace std;
/*-----*/
typedef vector<double> dvector;
typedef vector<vector<double> > dmatrix;
/*-----*/
dvector operator +(dvector A, dvector B){
int N = A.size();
dvector C(N);
```

```

for (int i=0;i<N;i++) { C[i]=A[i]+B[i]; }
return(C);
}

```

```

dvector operator -(dvector A, dvector B){
int N = A.size();
dvector C(N);
for (int i=0;i<N;i++) { C[i]=A[i]-B[i]; }
return(C);
}

```

```

dvector operator *(double A, dvector B){
int N = B.size();
dvector C(N);
for (int i=0;i<N;i++) { C[i]=A*B[i]; }
return(C);
}

```

```

dvector operator *(dvector A, double B){
int N = A.size();
dvector C(N);
for (int i=0;i<N;i++) { C[i]=A[i]*B; }
return(C);
}

```

```

double operator *(dvector A, dvector B){
int N = A.size();
double D;
D =0.0;
for (int i=0;i<N;i++) { D=D+A[i]*B[i]; }
return(D);
}

```

```

/*-----*/

```

```

dmatrix operator +(dmatrix A, dmatrix B){
int N = A.size();
dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A[i]+B[i]; }
return(C);
}

```

```

dmatrix operator -(dmatrix A, dmatrix B){
int N = A.size();

```

```

dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A[i]-B[i]; }
return(C);
}

```

```

dmatrix operator *(double A, dmatrix B){
int N = B.size();
dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A*B[i]; }
return(C);
}

```

```

dmatrix operator *(dmatrix A, double B){
int N = A.size();
dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A[i]*B; }
return(C);
}

```

```

/*-----*/
#endif

```

Що стосується методу Адамса-Башфорта [2] чисельного розв'язку задачі Коші для систем диференціальних рівнянь, то він є значно точнішим за метод Рунге-Кути, але разом з тим і складнішим в реалізації його програмного алгоритму. Справа в тому, що метод Рунге-Кути є однокроковим, а метод Адамса-Башфорта — багатокроковим методом чисельного розв'язку задачі Коші для систем диференціальних рівнянь. Тому алгоритм для методу Рунге-Кути розвиває ітераційний процес із заданих початкових умов ($i = 0$)

$$\vec{r}_{i+1} = \vec{r}_i + \frac{\Delta t}{6}(\vec{k}_1^i + 2\vec{k}_2^i + 2\vec{k}_3^i + \vec{k}_4^i)$$

$$\begin{cases} \vec{k}_1^i = \vec{f}(\vec{r}_i, t_i), \\ \vec{k}_2^i = \vec{f}\left(\vec{r}_i + \frac{\Delta t}{2}\vec{k}_1^i, t_i + \frac{\Delta t}{2}\right), \\ \vec{k}_3^i = \vec{f}\left(\vec{r}_i + \frac{\Delta t}{2}\vec{k}_2^i, t_i + \frac{\Delta t}{2}\right), \\ \vec{k}_4^i = \vec{f}\left(\vec{r}_i + \Delta t\vec{k}_3^i, t_i + \Delta t\right), \end{cases}$$

де Δt — крок зміщення вздовж часової осі t . Сумарна помилка обчислень при цьому складає $O(\Delta t^4)$. В той же час метод Адамса-Башфорта є

багатокроковим з сумарною помилкою обчислень $O(\Delta t^5)$

$$\vec{r}_{i+1} = \vec{r}_i + \frac{\Delta t}{24}(55\vec{f}(\vec{r}_i, t_i) - 59\vec{f}(\vec{r}_{i-1}, t_{i-1}) + 37\vec{f}(\vec{r}_{i-2}, t_{i-2}) - 9\vec{f}(\vec{r}_{i-3}, t_{i-3}))$$

а тому і несамодостатнім. Проблема в тому, що при розвитку ітераційного процесу, починаючи з початкових умов, коли $i = 0$, у нас виникає недостача даних при $i = -1, -2, -3$. Тому в даному випадку для вирішення цієї проблеми, зазвичай, для розрахунку значень невідомих функцій в перших 4-х точках (коли $i = 0, 1, 2, 3$) використовують однокрокові методи типу Рунге-Кути, а потім вже переходять до методу Адамса-Башфорта [2]. Оскільки використання методу Адамса-Башфорта для розв'язку задачі Коші додатково вимагає застосування однокрокових методів, то виглядає раціональною побудова універсального класу для розв'язку задачі Коші для довільної системи диференціальних рівнянь 1-го порядку 4-порядку точності. В якості однокрокового методу тут доцільно вибрати метод Рунге-Кути.

Програмування класу **ClsDES4** суттєво спрощує використання таких типів даних як **dvector**, **dmatrix** та перезавантажених операцій. Для прикладу лістинг класу **ClsDES4** приведено нижче.

```
// ClsDES4.h -----
#ifndef ClsDES4H
#define ClsDES4H
/*-----*/
#include <vector>
#include <Math.hpp>
#include <math.h>
#include "Matrix0.h"
/*-----*/
class ClsDES4
{
private:
double t0,t1,dt;
dvector Q0;
dvector (*F)(dvector Q,double t);
public:
ClsDES4(dvector (*_F)(dvector _Q,double _t),dvector _Q0,
double _t0,double _t1,double _dt) :
F(_F),Q0(_Q0),t0(_t0),t1(_t1),dt(_dt) {};

ClsDES4(ClsDES4 &Z) : F(Z.F),Q0(Z.Q0),t0(Z.t0),t1(Z.t1),dt(Z.dt) {};
```

```

dvector &Y0(){return Q0;}
double &T0(){return t0;}
double &T1(){return t1;}
double &dT(){return dt;}

dmatrix fn_RK4();
dmatrix fn_AB4();
};

dmatrix ClsDES4::fn_RK4() {
int N = (T1()-T0())/dT();
dmatrix Y(N+1,1);
Y[0]=Y0();
for(int i=0;i<N;i++) {
dvector K1=F(Y[i], T0()+i*dT());
dvector K2=F(Y[i]+(dT()/2)*K1, T0()+(i+0.5)*dT());
dvector K3=F(Y[i]+(dT()/4)*(K1+K2), T0()+(i+0.5)*dT());
dvector K4=F(Y[i]+dT()*((-1)*K2+2*K3), T0()+(i+1)*dT());
Y[i+1]=Y[i]+(dT()/6)*(K1+4*K3+K4);
}
return Y;
};

dmatrix ClsDES4::fn_AB4() {
int N = (T1()-T0())/dT();
dmatrix Y(N+1,1);
ClsDES4 obj=*this;
dmatrix tmpY=obj.fn_RK4();
for(int i=0;i<4;i++) Y[i]=tmpY[i];

for(int i=3;i<N;i++) {
Y[i+1]=Y[i]+(dT()/24)*
(55*F(Y[i], T0()+i*dT())-
59*F(Y[i-1],T0()+(i-1)*dT())+
37*F(Y[i-2],T0()+(i-2)*dT())-
9*F(Y[i-3],T0()+(i-3)*dT())
);
};
return Y;
};
/*-----*/
#endif

```

Звернемо увагу на те, що в класі **ClsDES4** визначені лише прототипи функцій **fn_RK4()** та **fn_AB4()**, а їх реалізація винесена за межі класу. При цьому у функції **fn_AB4()** створюється об'єкт класу **obj**, а потім, за його посередництвом, викликається метод класу **tmpY=obj.fn_RK4()**, який дозволяє визначити перші 4 значення невідомих функцій які необхідні для розвитку алгоритму Адамса-Башфорта.

З метою тестування класу **ClsDES4** для розв'язку системи диференціальних рівнянь використаємо простий приклад динаміки гармонічного осцилятора з дисипативним елементом під дією зовнішньої сили

$$\begin{cases} \frac{dZ}{d\tau} + 2\beta Z + Y = F_0 \cos(\Omega\tau) \\ \frac{dY}{d\tau} = Z \end{cases}$$

при заданих початкових умовах.

6.2. Рекомендації щодо виконання завдання

Приклад головної форми додатку приведено на Рис.6.1 та Рис.6.2 (див. далі). Там же приведені вхідні дані задачі, що задаються по замовчуванні. Для її побудови необхідно використати такі компоненти

$$\begin{aligned} TForm &- 1 \text{ шт.}; & TPanel &- 2 \text{ шт.}; \\ TRadioButton &- 4 \text{ шт.}; & TGroupBox &- 2 \text{ шт.}; \\ TEdit &- 11 \text{ шт.}; & TLabel &- 11 \text{ шт.}; & TButton &- 3 \text{ шт.}; \\ TStringGrid &- 1 \text{ шт.}; & TImage &- 1 \text{ шт.}; \end{aligned} \quad (6.1)$$

які знаходяться на інструментальній панелі програмного середовища.

В головному файлі проекту знаходиться програмний код, який реалізує інтерфейс додатку (див. Рис.6.1 та Рис.6.2) та його функціонал.

Приклад програмного коду головного файлу, що реалізує інтерфейс та функціонал додатку, можна написати у такому вигляді

```
/*- Unit01.cpp -----*/
#include <vcl.h>
#pragma hdrstop
#include <vector>
#include <Math.hpp>
#include <math.h>
using namespace std;
#include "Unit01.h"
```

```

#include "ClsDES4.h"
/*-----*/
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmUnit01 *frmUnit01;

double B,Af,Wf,W1,W2,T0,T1,DT,Cq,Ct,dT,dQ,dV,dZ;
int flag,i,NR,NC;
dmatrix resDES;

dvector fn_Force0(dvector Q, double t) {
int N = Q.size();
dvector F(N);
F[0] = Q[1];
F[1] = -2*B*Q[1]-Q[0]+Af*cos(Wf*t);
return F;
}

TRect aRect;
/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner)
: TForm(Owner)
{
}
/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)
{
RadioButton2Click((TObject *)Sender);
flag = (RadioButton0->Checked) ? 0 : 1;
B = StrToFloat(Edit0->Text);
Af = StrToFloat(Edit1->Text);
Wf = StrToFloat(Edit2->Text);
dvector fn_Data0(2);
fn_Data0[0] = StrToFloat(Edit5->Text);
fn_Data0[1] = StrToFloat(Edit6->Text);
T0 = StrToFloat(Edit3->Text);
T1 = StrToFloat(Edit4->Text);
DT = StrToFloat(Edit7->Text);

W1 = (1-B*B > 0) ? sqrt(1-B*B) : 0;
W2 = (1-2*B*B > 0) ? sqrt(1-2*B*B) : 0;
Edit11->Text = FloatToStr(SimpleRoundTo(W2,-8));

```

```

ClsDES4 DESolve(fn_Force0, fn_Data0, T0, T1, DT);
resDES = DESolve.fn_AB4();

    NR = (T1-T0)/DT;
    NC = fn_Data0.size();
    StringGrid0->RowCount = NR+1;
    StringGrid0->ColCount = NC+1;
    StringGrid0->Cells[1][0] = " Відхилення ";
    StringGrid0->Cells[2][0] = " Швидкість ";

    for(int i=0;i<NR;i++) {
StringGrid0->Cells[0][i] = " "+FloatToStr((i-1)*DT)+" ";
for(int j=0;j<NC;j++) {
StringGrid0->Cells[j+1][i+1] = resDES[i][j];
}
}

    StringGrid0->Cells[0][0] = " Час ";

    Cq = Image0->Height/2;
    Ct = 25;
    Button1Click((TObject *)Sender);
}
/*-----*/
void __fastcall TfrmUnit01::RadioButton0Click(TObject *Sender)
{
flag = (RadioButton0->Checked) ? 0 : 1;
Button0Click((TObject *)Sender);
}
/*-----*/
void __fastcall TfrmUnit01::RadioButton2Click(TObject *Sender)
{
StringGrid0->Visible = RadioButton2->Checked;
Image0->Visible = RadioButton3->Checked;
}
/*-----*/
void __fastcall TfrmUnit01::Button0Click(TObject *Sender)
{
Button1Click((TObject *)Sender);
FormShow((TObject *)Sender);
}

```

```

flag = (RadioButton0->Checked) ? 0 : 1;
i = 0;
Timer0->Enabled = true;
}
/*-----*/
void __fastcall TfrmUnit01::Timer0Timer(TObject *Sender)
{
while (i <= NR) {
dT = i*DT*8;
dQ = -resDES[i][0]*5e0;
dV = -resDES[i][1]*5e0;
if (i < 1) {
Image0->Canvas->Pen->Color=clWhite;
Image0->Canvas->LineTo(Ct,Cq);
} else {
Image0->Canvas->Pen->Color=(flag == 0) ? clBlue : clRed;
dZ = (flag == 0) ? dQ : dV;
Image0->Canvas->LineTo(Ct+dT,Cq+dZ);
}
i++;
}
}
/*-----*/
void __fastcall TfrmUnit01::Button1Click(TObject *Sender)
{
Timer0->Enabled = false;
Image0->Canvas->Pen->Width=5;
Image0->Canvas->Pen->Color=clWhite;
aRect = Rect(0,0,ClientWidth,ClientHeight);
Image0->Canvas->FillRect(aRect);
Image0->Canvas->FillRect(Rect(0,0,Image0->Width,Image0->Height));
Image0->Canvas->MoveTo(Ct,Cq);
}
/*-----*/
void __fastcall TfrmUnit01::Button2Click(TObject *Sender)
{
frmUnit01->Close();
}
/*-----*/

```

В результаті роботи програми ми отримаємо матрицю **resDES[N][2]**

$\left(N = \frac{T1 - T0}{DT}\right)$, якою можна розпорядитися відповідно до мети проекту.

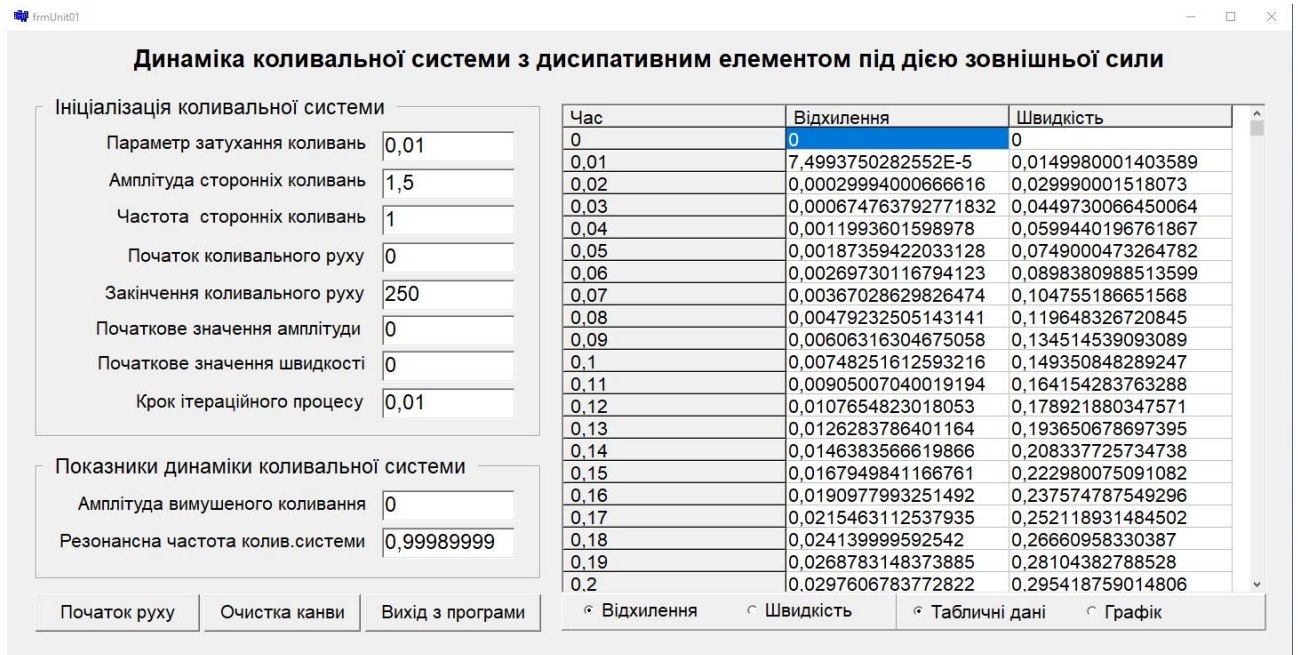


Рис. 6.1. Додаток для дослідження динаміки коливальної системи (Таблична візуалізація результатів розрахунку)

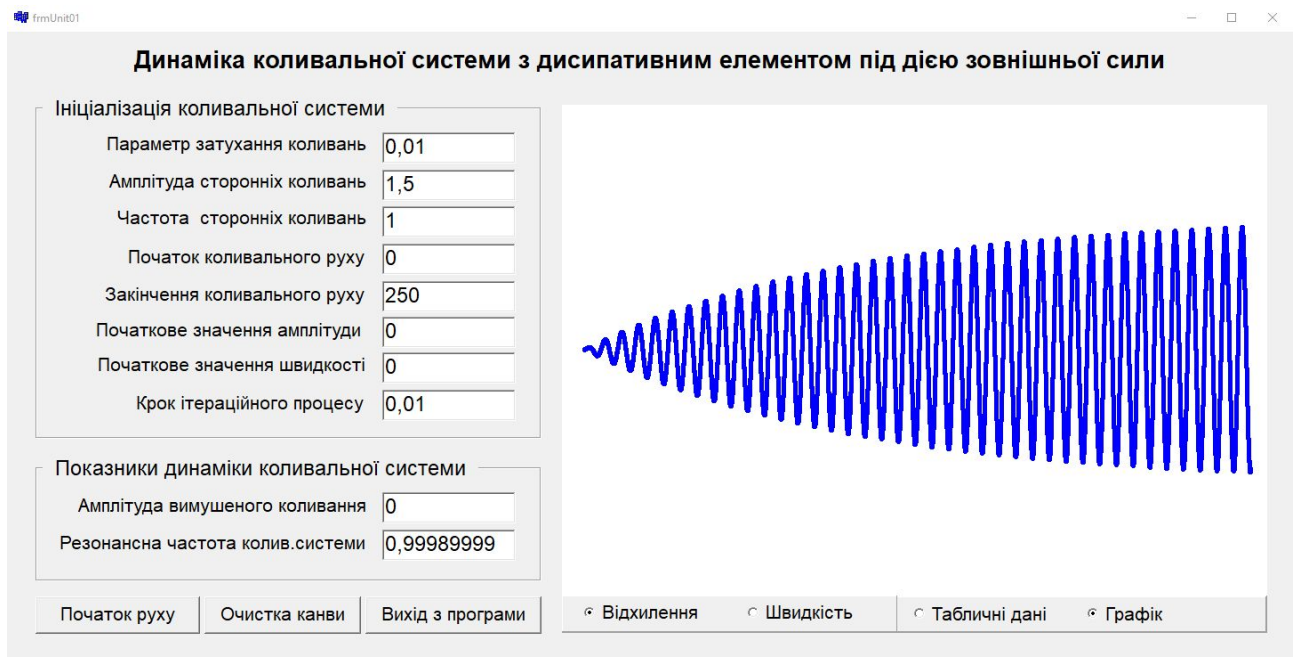


Рис. 6.2. Додаток для дослідження динаміки коливальної системи (Графічна візуалізація результатів розрахунку)

6.3. Варіанти завдань до комп'ютерного практикуму

- Параметр затухання коливань: [0.01; 0.0125; 0.05; 0.075; 0.02;]
- Амплітуда коливань зовнішньої сили: [1.25; 1.5; 1.75; 2; 2.25;]
- Частота коливань зовнішньої сили: [1; 1.25; 1.5; 1.75; 2;]

- Початок коливального руху: [0]
- Закінчення коливального руху: [225; 250; 275; 280; 285;]
- Крок ітераційного процесу: [0.005; 0.0075; 0.01; 0.0125; 0.015;]

6.4. Підготовка до комп'ютерного практикуму

- Розглянути теоретичний матеріал щодо використання контейнерних типів даних для програмування розв'язку задачі Коші методом Адамса-Башфорта 4-порядку.
- Ознайомитися з теоретичними відомостями щодо теми комп'ютерного практикуму.
- Ознайомитися з рекомендаціями щодо виконання завдання.

6.5. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.
- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project06.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project05.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям щодо виконання роботи, аналогічно Рис.6.1 та Рис.6.2, розташуйте на головній формі проекту колекцію компонентів (6.1), та надайте їх відповідне оформлення.
- У заголовочному файлі **Matrix0.h** підключіть бібліотеки: `#include <vector>`.
- У заголовочному файлі **ClsDES4.h** підключіть заголовочний файл: `#include "Matrix0.h"`.

- У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "ClsDES4.h"`.
- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).
- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Виконайте тестові розрахунки на прикладах з відомою відповіддю.

6.6. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості.

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочних файлів **Matrix0.h**, **ClsDES4.h** та головного файлу **Unit01.cpp**;
 - висновки.

6.7. Контрольні запитання

- Як означаються контейнерні типи даних **dvector** та **dmatrix**?
- У чому заключаються переваги використання класів для розв'язку задачі Коші довільної системи диференціальних рівнянь 1-го порядку у порівнянні з структурним програмуванням?
- У чому полягає метод перезавантаження операцій для контейнерних типів даних **dvector** та **dmatrix**?
- У чому заключається особливість методу Адамса-Башфорта в порівнянні з методом Рунге-Кути для розв'язку задачі Коші?

- Для якого типу диференціальних рівнянь метод Адамса-Башфорта дає кращі результати в порівнянні з методом Рунге-Кути?

7. Комп'ютерний практикум №7. Програмування додатку для дослідження динаміки коливальних систем із дисипативним елементом з використанням алгоритму Адамса-Башфорта

Мета роботи: Отримати досвід використання класів з використанням контейнерних типів даних мови програмування C++ для програмування розв'язку задачі Коші для довільної системи диференціальних рівнянь 1-го порядку у випадку немонотонних динамічних систем.

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] додаток для розв'язків задачі Коші з використанням класу **ClsDES4** для дослідження різних режимів динаміки гармонічного осцилятора під дією зовнішньої сили та параметричної коливальної системи.

Вхідні параметрами задачі:

- Параметр затухання коливань
- Амплітуда коливань зовнішнього впливу
- Частота коливань зовнішнього впливу
- Початкове значення амплітуда коливань осцилятора
- Початкове значення швидкості коливань осцилятора
- Початок коливального руху
- Закінчення коливального руху
- Крок ітераційного процесу

Розрахункові параметрами задачі:

- Резонансна частота осцилятора
- Амплітуда зміщень вимушених коливань осцилятора
- Амплітуда швидкості вимушених коливань осцилятора

7.1. Теоретичні відомості

В процесі виконання завдання комп'ютерного практикуму потрібно створити додаток для дослідження динаміки

- гармонічного осцилятора з дисипативним елементом, на який діє узальнена періодична сила при заданих початкових умовах (див. Рис.7.1);
- параметричної коливальної системи з дисипативним елементом (див. Рис.7.2).

Математична модель гармонічного осцилятора з дисипативним елементом, на який діє узальнена періодична сила описує широкий клас фізичних явищ. До таких систем, зокрема, відносяться

- Вантаж прикріплений до кінця пружини динаміка якого описується 2-м законом Ньютона [7]

$$m \frac{d^2 x}{dt^2} + \alpha \frac{dx}{dt} + kx = f_0 \cos(\omega t),$$

x — координата центру маси вантажу, m — маса вантажу, α — коефіцієнт тертя, k — коефіцієнт жорсткості пружини, f_0 — амплітуда механічної сили, ω — циклічна частота механічної сили.

- Процеси, які відбуваються в послідовному коливальному контурі, складеному із послідовно з'єднаних резистора R , конденсатора C і котушки індуктивності L , на який подано напругу $U(t)$. Динаміка електричного заряду в такому контурі описується рівнянням [8]:

$$L \frac{d^2 q}{dt^2} + R \frac{dq}{dt} + \frac{1}{C} q = U_0 \cos(\omega t)$$

де q — заряд, що проходить по ланцюгу, $I = \frac{dq}{dt}$ — сила електричного струму. При цьому реалізується електромеханічна аналогія "сила-напруга".

- Процеси, які відбуваються в паралельному коливальному контурі, який складається із резистора R , конденсатора C і котушки індуктивності L , з'єднаних паралельно і підключених до джерела струму I . Динаміка напруги в такому контурі описується рівнянням [8]:

$$C \frac{d^2 U}{dt^2} + \frac{1}{R} \frac{dU}{dt} + \frac{1}{L} U = \omega I_0 \cos(\omega t)$$

де U – напруга на резисторі, C – ємність та L – індуктивність, I_0 – амплітуда сили зовнішнього струму. При цьому реалізується електромеханічна аналогія "сила-струм".

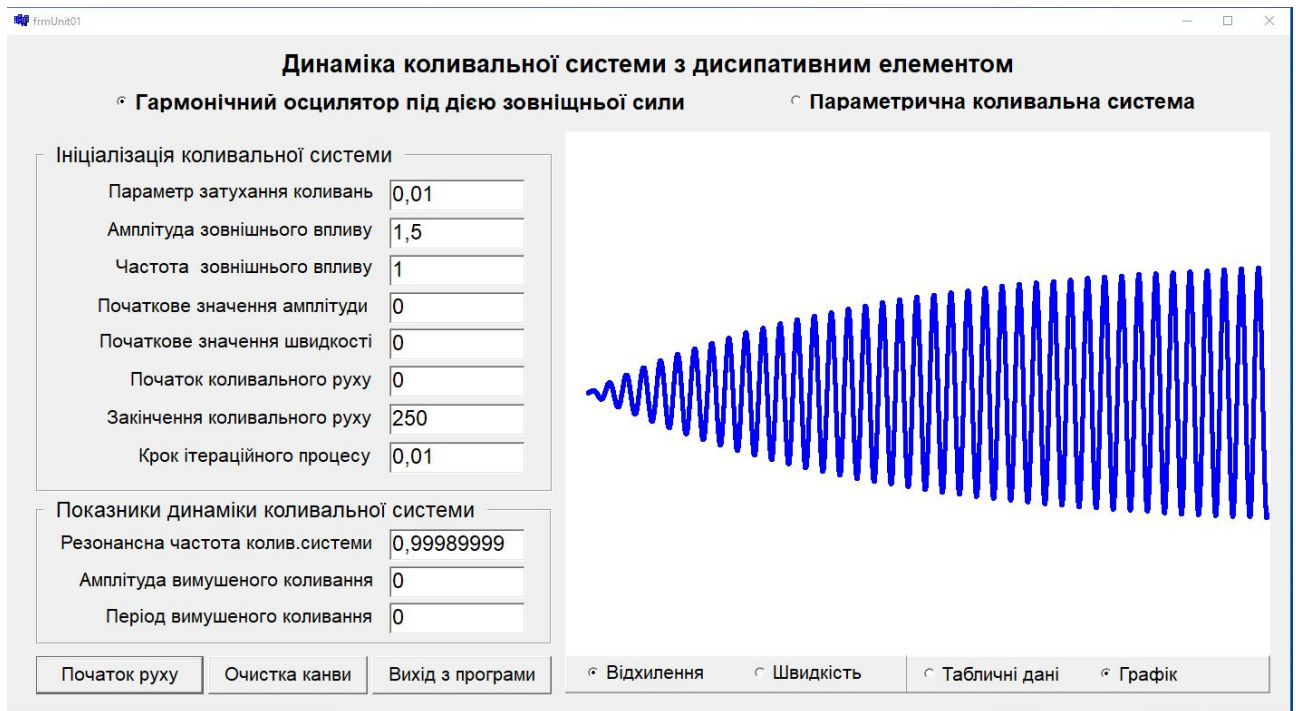


Рис. 7.1. Візуальний додаток для дослідження динаміки коливальної системи (Діє зовнішня сила)

Всі ці рівняння за рахунок простих алгебраїчних перетворень можна перезаписати в наступному вигляді

$$\frac{d^2 y}{dt^2} + 2\alpha \frac{dy}{dt} + \omega_0^2 y = g_0 \cos(\omega t).$$

Для програмування чисельного розв'язку отриманого рівняння його необхідно звести до системи двох диференціальних рівнянь першого порядку, а саме

$$\begin{cases} \frac{dz}{dt} + 2\alpha z + \omega_0^2 y = g_0 \cos(\omega t) \\ \frac{dy}{dt} = z \end{cases}$$

де α — параметер затухання, ω_0 — циклічна частота системи, g_0 — амплітуда зовнішньої сили, ω — циклічна частота зовнішньої сили.

Отриману систему рівнянь потрібно привести до безрозмірного вигляду за рахунок нових змінних, а саме

$$Y = \frac{y}{y_0}, \quad Z = \frac{z}{z_0}, \quad \tau = \frac{t}{t_0} = t\omega_0, \quad \beta = \frac{\alpha}{\omega_0}, \quad F_0 = \frac{g_0}{y_0\omega_0}, \quad \Omega = \frac{\omega}{\omega_0},$$

де має місце зв'язок $y_0 = z_0 t_0$.

В нових змінних система рівнянь набуває форму зручну для програмування її чисельного розв'язку

$$\begin{cases} \frac{dZ}{d\tau} + 2\beta Z + Y = F_0 \cos(\Omega\tau) \\ \frac{dY}{d\tau} = Z \end{cases}$$

Систему рівнянь доповнимо початковими умовами $Y(\tau = 0) = 0$ $Z(\tau = 0) = Z_0$. Що стосується величин β , F_0 , Ω то їх значення будуть задаватися в програмі чисельного розв'язку останньої системи рівнянь.

Коливальна система на яку не діє зовнішня сила, але коефіцієнт пружності якої змінюється з часом по періодичному закону називається параметричною коливальною системою [7]. Як і в попередньому випадку множину параметричних задач можна описати одним і тим же диференціальним рівнянням, яке в даному випадку буде мати вигляд

$$\frac{d^2 y}{dt^2} + 2\alpha \frac{dy}{dt} + (\omega_0^2 - 2q \sin(2\omega t))y = 0.$$

Для програмування чисельного розв'язку отриманого рівняння його також необхідно звести до системи двох диференціальних рівнянь першого порядку, а саме

$$\begin{cases} \frac{dz}{dt} + 2\alpha z + (\omega_0^2 - 2q \sin(2\omega t))y = 0 \\ \frac{dy}{dt} = z \end{cases}$$

де α — параметер затухання, ω_0 — циклічна частота системи, q — амплітуда параметричного впливу, 2ω — циклічна частота параметричного впливу.

Отриману систему рівнянь потрібно привести до безрозмірного вигляду за рахунок нових змінних, а саме

$$Y = \frac{y}{y_0}, \quad Z = \frac{z}{z_0}, \quad \tau = \frac{t}{t_0} = t\omega_0, \quad \beta = \frac{\alpha}{\omega_0}, \quad \lambda = \frac{q}{\omega_0^2}, \quad \Omega = \frac{\omega}{\omega_0},$$

де має місце зв'язок $y_0 = z_0 t_0$.

В нових змінних система рівнянь набуває форму зручну для програмування її чисельного розв'язку

$$\begin{cases} \frac{dZ}{d\tau} + 2\beta Z + (1 - 2\lambda \sin(2\Omega\tau))Y = 0 \\ \frac{dY}{d\tau} = Z \end{cases}$$

Легко зрозуміти, що параметричний резонанс, на відміну від силового, може відбуватися тільки за ненульових початкових умовах, оскільки завдання $Y(\tau = 0) = 0$, $Z(\tau = 0) = 0$ приводить, очевидно, до тотожно нульового рішення (рівняння однорідні).

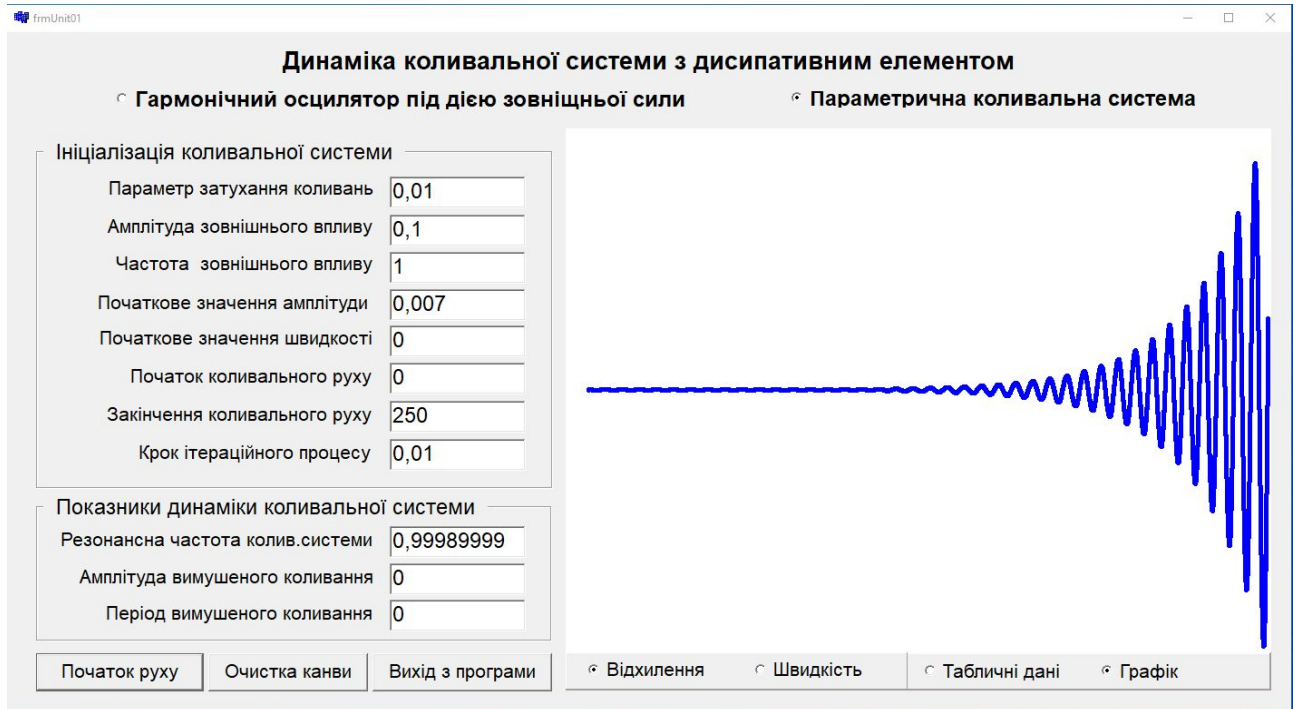


Рис. 7.2. Візуальний додаток для дослідження динаміки коливальної системи (Режим параметричних коливань)

7.2. Рекомендації щодо виконання завдання

Звернемо увагу на те, що внаслідок використання класу **ClsDES4** програмни код для аналізу динаміки як гармонічного осцилятора та і параметричної коливальної системи спільний. Досить тільки змінювати за допомогою компонента **TRadioButton** вираз для силової функції у відповідному місці програмного крду.

Приклад головної форми додатку приведено на Рис.7.1 та Рис.7.2. Там же приведені вхідні дані задачі, що задаються по замовчуванні. Для її побудови необхідно використати такі компоненти

$$\begin{aligned}
 TForm & - 1 \text{ шт.}; & TPanel & - 5 \text{ шт.}; \\
 TRadioButton & - 6 \text{ шт.}; & TGroupBox & - 2 \text{ шт.}; \\
 TEdit & - 11 \text{ шт.}; & TLabel & - 11 \text{ шт.}; & TButton & - 3 \text{ шт.}; \\
 TStringGrid & - 1 \text{ шт.}; & TImage & - 1 \text{ шт.};
 \end{aligned}
 \tag{7.1}$$

які знаходяться на інструментальній панелі програмного середовища.

В головному файлі проекту знаходиться програмний код, який реалізує інтерфейс додатку (див. Рис.7.1 та Рис.7.2) та його функціонал. При його

написанні використайте раніше створені вами заголовочні файли **Matrix0.h** та **ClsDES4.h**.

Програмний код головного файлу, що реалізує інтерфейс та функціонал додатку, можна написати у такому вигляді

```

/*- Unit01.cpp -----*/
#include <vcl.h>
#pragma hdrstop
#include <vector>
#include <Math.hpp>
#include <math.h>
using namespace std;
#include "Unit01.h"
#include "ClsDES4.h"
/*-----*/
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmUnit01 *frmUnit01;

double B,Af,Wf,W1,W2,T0,T1,DT,Cq,Ct,dT,dQ,dV,dZ;
int flag,i,NR,NC;
dmatrix resDES;

dvector fn_Force0(dvector Q, double t) {
int N = Q.size();
dvector F(N);
F[0] = Q[1];
(RadioForce0->Checked) ?
F[1] = -2*B*Q[1]-Q[0]+Af*cos(Wf*t):
F[1] = -2*B*Q[1]-(1+2*Af*sin(2*Wf*t))*Q[0];
return F;
}

TRect aRect;
/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner)
: TForm(Owner)
{
}
/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)

```

```

{
RadioButton2Click((TObject *)Sender);
flag = (RadioButton0->Checked) ? 0 : 1;
B = StrToFloat(Edit0->Text);
Af = StrToFloat(Edit1->Text);
Wf = StrToFloat(Edit2->Text);
dvector fn_Data0(2);
fn_Data0[0] = StrToFloat(Edit5->Text);
fn_Data0[1] = StrToFloat(Edit6->Text);
T0 = StrToFloat(Edit3->Text);
T1 = StrToFloat(Edit4->Text);
DT = StrToFloat(Edit7->Text);

W1 = (1-B*B > 0) ? sqrt(1-B*B) : 0;
W2 = (1-2*B*B > 0) ? sqrt(1-2*B*B) : 0;
Edit11->Text = FloatToStr(SimpleRoundTo(W2,-8));

ClsDES4 DESolve(fn_Force0, fn_Data0, T0, T1, DT);
resDES = DESolve.fn_AB4();

    NR = (T1-T0)/DT;
    NC = fn_Data0.size();
    StringGrid0->RowCount = NR+1;
    StringGrid0->ColCount = NC+1;
    StringGrid0->Cells[1][0] = " Відхилення ";
    StringGrid0->Cells[2][0] = " Швидкість ";

    for(int i=0;i<NR;i++) {
StringGrid0->Cells[0][i] = " "+FloatToStr((i-1)*DT)+" ";
for(int j=0;j<NC;j++) {
StringGrid0->Cells[j+1][i+1] = resDES[i][j];
}
}

    StringGrid0->Cells[0][0] = " Час ";

    Cq = Image0->Height/2;
    Ct = 25;
    Button1Click((TObject *)Sender);
}
/*-----*/
void __fastcall TfrmUnit01::RadioButton0Click(TObject *Sender)

```

```

{
flag = (RadioButton0->Checked) ? 0 : 1;
Button0Click((TObject *)Sender);
}
/*-----*/
void __fastcall TfrmUnit01::RadioButton2Click(TObject *Sender)
{
StringGrid0->Visible = RadioButton2->Checked;
Image0->Visible = RadioButton3->Checked;
}
/*-----*/
void __fastcall TfrmUnit01::Button0Click(TObject *Sender)
{
Button1Click((TObject *)Sender);
FormShow((TObject *)Sender);
flag = (RadioButton0->Checked) ? 0 : 1;
i = 0;
Timer0->Enabled = true;
}
/*-----*/
void __fastcall TfrmUnit01::Timer0Timer(TObject *Sender)
{
while (i <= NR) {
dT = i*DT*8;
dQ = -resDES[i][0]*5e0;
dV = -resDES[i][1]*5e0;
if (i < 1) {
Image0->Canvas->Pen->Color=clWhite;
Image0->Canvas->LineTo(Ct,Cq);
} else {
Image0->Canvas->Pen->Color=(flag == 0) ? clBlue : clRed;
dZ = (flag == 0) ? dQ : dV;
Image0->Canvas->LineTo(Ct+dT,Cq+dZ);
}
i++;
}
}
/*-----*/
void __fastcall TfrmUnit01::Button1Click(TObject *Sender)
{
Timer0->Enabled = false;
}

```

```

Image0->Canvas->Pen->Width=5;
Image0->Canvas->Pen->Color=clWhite;
aRect = Rect(0,0,ClientWidth,ClientHeight);
Image0->Canvas->FillRect(aRect);
Image0->Canvas->FillRect(Rect(0,0,Image0->Width,Image0->Height));
Image0->Canvas->MoveTo(Ct,Cq);
}
/*-----*/
void __fastcall TfrmUnit01::Button2Click(TObject *Sender)
{
frmUnit01->Close();
}
/*-----*/

```

В результаті роботи додатку ми отримаємо графічні залежності динамічної змінної та її швидкості від часу при заданих параметрах задачі та певних початкових умовах. Суттєво, що відповідальним за розв’язок задачі Коші методом Адамса-Башфорта 4-порядку є клас **ClsDES4**.

7.3. Варіанти завдань до комп’ютерного практикуму

- Параметр затухання коливань: [0.01; 0.0125; 0.05; 0.075; 0.02;]
- Амплітуда коливань зовнішньої сили: [1.25; 1.5; 1.75; 2; 2.25;]
- Частота коливань зовнішньої сили: [1; 1.25; 1.5; 1.75; 2;]
- Початкове значення амплітуда коливань осцилятора: [0.005; 0.007; 0.009; 0.01; 0.012;]
- Початкове значення швидкості коливань осцилятора: [0]
- Початок коливального руху: [0]
- Закінчення коливального руху: [225; 250; 275; 280; 285;]
- Крок ітераційного процесу: [0.005; 0.0075; 0.01; 0.0125; 0.015;]

7.4. Підготовка до комп’ютерного практикуму

- Розглянути теоретичний матеріал щодо використання класу **ClsDES4** для розв’язку задачі Коші методом Адамса-Башфорта 4-порядку.
- Ознайомитися з теоретичними відомостями щодо теми комп’ютерного практикуму.

- Ознайомитися з рекомендаціями щодо виконання завдання.

7.5. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.
- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project07.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project06.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям щодо виконання роботи, аналогічно Рис.7.1 та Рис.7.2, розташуйте на головній формі проекту колекцію компонентів (7.1) та надайте їх відповідне оформлення.
- У заголовочному файлі **Matrix0.h** підключіть бібліотеки: `#include <vector>`.
- У заголовочному файлі **ClsDES4.h** підключіть заголовочний файл: `#include "Matrix0.h"`.
- У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "ClsDES4.h"`.
- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).
- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Проведіть дослідження впливу значень параметрів задачі та початкових умов на динаміку коливальної системи. Для цього проаналізуйте

динаміку коливальної системи при

- $\beta \in \{0, 0.025, 0.050, 0.075, 0.1\}$;
- $\Omega \in \{\Omega_{res}, \Omega_{res} \pm 0.025, \Omega_{res} \pm 0.050, \Omega_{res} \pm 0.075, \Omega_{res} \pm 0.1\}$;
- $Y_0 = 0, Z_0 = 0$; $Y_0 = 1, Z_0 = 0$; $Y_0 = 0, Z_0 = 1$;

7.6. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості.

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочних файлів **Matrix0.h**, **ClsDES4.h** та головного файлу **Unit01.cpp**;
 - графічні залежності динамічних координат та їх швидкості від часу при заданих значеннях параметрів задачі та початкових умов.
 - аналіз отриманих результатів та висновки.

7.7. Контрольні запитання

- Чому при розв'язку задачі Коші методами обчислювальної математики для немонотонних динамічних систем алгоритм Адамса-Башфорта дає значно кращі результати у порівнянні з алгоритмом Рунге-Кути?
- Що таке явище резонансу у динаміці коливальної системи?
- Як впливають на динаміку коливальної системи початкові умови задачі?
- Як впливає на динаміку коливальної системи зростання параметра затухання?
- Як впливає на динаміку вимушених коливань амплітуда зовнішньої сили?

8. Комп'ютерний практикум №8. Програмування додатку "Матрична алгебра (Основні операції матричної алгебри)" на основі об'єктного підходу та дослідження його функціоналу

Мета роботи: Отримати досвід використання об'єктного підходу та перезавантаження матричних операцій при програмуванні додатку для виконання операцій матричної алгебри [2].

Створити у програмному середовищі Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 [5,6] додаток для виконання операцій матричної алгебри.

Вхідні параметрами задачі:

- Матриця $\hat{A}$ довільної розмірності;
- Матриця $\hat{B}$ розмірність якої узгоджена з розмірністю матриці $\hat{A}$;
- Коефіцієнти лінійної комбінації матриць.

Вихідна матриця C : Результат опрацювання в додатку матриць $\hat{A}$ та $\hat{B}$ відповідно до базових операцій матричної алгебри.

8.1. Теоретичні відомості

В процесі виконання завдання комп'ютерного практикуму потрібно створити додаток для виконання операцій матричної алгебри [2]

- Лінійна комбінація матриць $\hat{C} = \alpha \hat{A} + \beta \hat{B}$;
- Множення матриць $\hat{C} = \hat{A} * \hat{B}$;
- Розрахунок транспонованої матриці $\hat{C} = \hat{A}^+$;
- Розрахунок оберненої матриці $\hat{C} = \hat{A}^{-1}$;
- Ділення матриць $\hat{C} = \hat{A} * \hat{B}^{-1}$;
- Розрахунок псевдооберненої матриці $\hat{C} = (\hat{A}^+ * \hat{A})^{-1} * \hat{A}^+$.

8.2. Рекомендації щодо виконання завдання

Приклад головної форми додатку приведено на Рис.8.1. Там же приведені вхідні дані задачі, що задаються по замовчуванні. Для її побудови необхідно

використати такі компоненти

$$\begin{aligned}
 TForm &- 1 \text{ шт.}; & TPanel &- 5 \text{ шт.}; \\
 TEdit &- 7 \text{ шт.}; & TLabel &- 7 \text{ шт.}; \\
 TStringGrid &- 3 \text{ шт.}; & TButton &- 12 \text{ шт.};
 \end{aligned}
 \tag{8.1}$$

які знаходяться на інструментальній панелі програмного середовища. В

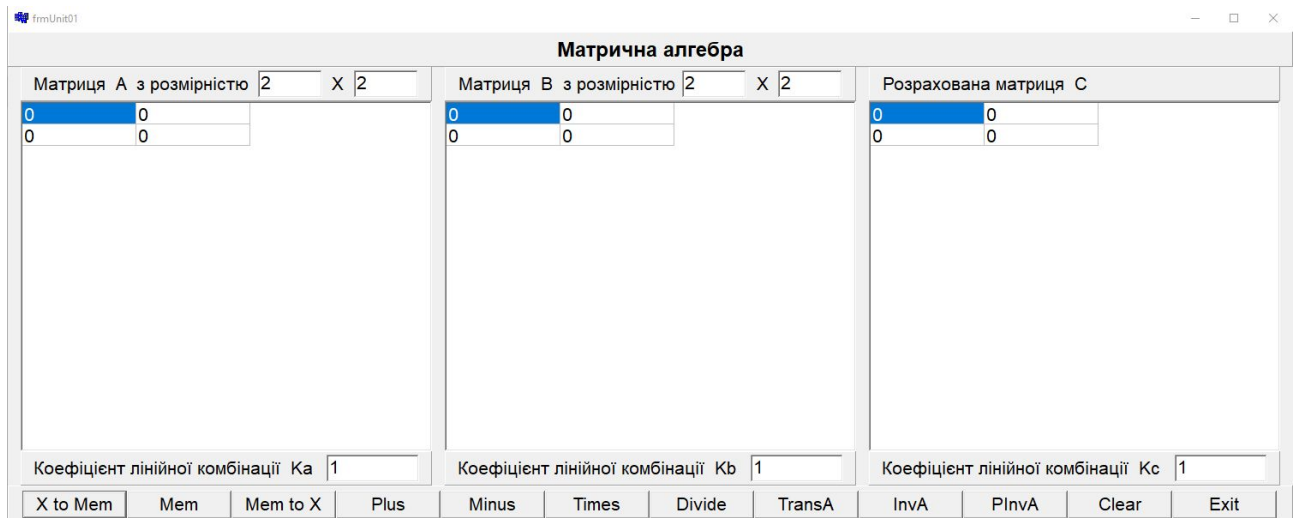


Рис. 8.1. Візуальний додаток для виконання операцій матричної алгебри

головному файлі проекту знаходиться програмний код, який реалізує інтерфейс додатку (див. Рис.8.1) та його функціонал. При його написанні потрібно буде запрограмувати та використати заголовочний файл **Matrix.h** в якому визначається клас **Matrix** для реалізації в повному обсязі матричної алгебри.

Варіант програмної реалізації класу **Matrix** має наступний вигляд

```

/*-- Matrix.h -----*/
#include <stdio.h>
#include <string.h>
#include <vector>
#include <Math.hpp>
#include <math.h>
using namespace std;
typedef vector<double> dvector;
typedef vector<vector<double> > dmatrix;
#define abs(a) (((a)>0)?(a):(-a))
/*-----*/
class Matrix
{
private:
bool flag;

```

```

int msize;
double *data;
public:
Matrix(int);
Matrix(const Matrix &);
Matrix operator *(const Matrix &);
void operator = (const Matrix &);
void Transpose();
Matrix Minor(int, int);
double Determinant();
Matrix Inverse();
void SetElement(int, int, double);
double GetElement(int, int) const;
~Matrix();
};
/*-----*/
Matrix :: ~Matrix()
{
delete [] data;
}
/*-----*/
Matrix Matrix :: operator *(const Matrix &a)
{
if(msize == a.msize)
{
double val = 0;
Matrix res(msize);
for(int i = 0; i < msize; i ++)
{
for(int j = 0; j < msize; j ++)
{
for(int k = 0; k < msize; k ++)
{
val += GetElement(k, i)*a.GetElement(j, k);
}
res.SetElement(j, i, val);
val = 0;
}
}
return res;
}
return Matrix(0);

```

```

}
/*-----*/
void Matrix :: operator = (const Matrix &m)
{
delete [] data;
data = new double[m.msize*m.msize];
msize = m.msize;
memcpy((void *)data, (void *)m.data, msize*msize*sizeof(double));
flag = m.flag;
}
/*-----*/
void Matrix :: Transpose()
{
flag = !flag;
}
/*-----*/
Matrix Matrix :: Inverse()
{
Matrix res(msize);
double Det = Determinant();
for(int i = 0; i < msize; i ++)
{
for(int j = 0; j < msize; j ++)
{
res.SetElement(i,j,Minor(j, i).
Determinant()/Det*((i+j)%2==0?-1.0:1.0));
}
}
return res;
}
/*-----*/
double Matrix :: Determinant()
{
if(msize == 1) return data[0];
else
{
double res = 0;
for(int i = 0; i < msize; i ++)
{
res += GetElement(i, 0)*(Minor(i, 0).
Determinant())*(i%2==0?-1.0:1.0);
}
}
}

```

```

}
return res;
}
}
/*-----*/
Matrix Matrix :: Minor(int i, int j)
{
if(i < msize && j < msize && i >= 0 && j >= 0)
{
if(msize > 1)
{
Matrix res(msize-1);
for(int i1 = 0; i1 < msize; i1 ++)
{
for(int j1 = 0; j1 < msize; j1 ++)
{
if(i1 != i && j1 != j)
{
int it = (i1 > i)?-1:0;
int jt = (j1 > j)?-1:0;
res.SetElement(i1 + it, j1 + jt, GetElement(i1,j1));
}
}
}
return res;
}
}
}
/*-----*/
Matrix :: Matrix(const Matrix &m) :
data(new double[m.msize*m.msize]), flag(m.flag)
{
msize = m.msize;
memcpy((void *)data, (void *)m.data, msize*msize*sizeof(double));
}
/*-----*/
Matrix :: Matrix(int n) :
msize(abs(n)), data(new double[n*n]),
flag(false)
{
memset((void *)data, 0, n*n*sizeof(double));

```

```

}
/*-----*/
void Matrix :: SetElement(int i, int j, double val)
{
if(i < msize && j < msize && i >= 0 && j >= 0)
{
if(flag)
data[j+i*msize] = val;
else
data[i+j*msize] = val;
}
}
/*-----*/
double Matrix :: GetElement(int i, int j) const
{
if(i < msize && j < msize && i >= 0 && j >= 0)
{
if(flag)
return data[j+i*msize];
else
return data[i+j*msize];
}
}
/*-----*/
/*-----*/
dvector operator +(dvector A, dvector B){
int N = A.size();
dvector C(N);
for (int i=0;i<N;i++) { C[i]=A[i]+B[i]; }
return(C);
}

dvector operator -(dvector A, dvector B){
int N = A.size();
dvector C(N);
for (int i=0;i<N;i++) { C[i]=A[i]-B[i]; }
return(C);
}

dvector operator *(double A, dvector B){
int N = B.size();

```

```

dvector C(N);
for (int i=0;i<N;i++) { C[i]=A*B[i]; }
return(C);
}

```

```

dvector operator *(dvector A, double B){
int N = A.size();
dvector C(N);
for (int i=0;i<N;i++) { C[i]=A[i]*B; }
return(C);
}

```

```

double operator *(dvector A, dvector B){
int N = A.size();
double D;
D =0.0;
for (int i=0;i<N;i++) { D=D+A[i]*B[i]; }
// D=SimpleRoundTo(sqrt(D),-8);
D=sqrt(D);
return(D);
}

```

```

/*-----*/
dmatrix operator +(dmatrix A, dmatrix B){
int N = A.size();
dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A[i]+B[i]; }
return(C);
}

```

```

dmatrix operator -(dmatrix A, dmatrix B){
int N = A.size();
dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A[i]-B[i]; }
return(C);
}

```

```

dmatrix operator *(double A, dmatrix B){
int N = B.size();
dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A*B[i]; }
return(C);
}

```

```
}
```

```
dmatrix operator *(dmatrix A, double B){
int N = A.size();
dmatrix C(N);
for (int i=0;i<N;i++) { C[i]=A[i]*B; }
return(C);
}
```

```
dmatrix operator *(dmatrix A, dmatrix B){
int AR = A.size();
int AC = A[0].size();
int BR = B.size();
int BC = B[0].size();
dmatrix C(AR,BC);
if (AC==BR) {
for (int i=0;i<AR;i++) {
for (int j=0;j<BC;j++) {
C[i][j] = 0;
for (int k=0;k<AC;k++) {
C[i][j] += A[i][k]*B[k][j];
}
}}
return(C);
}
else
{ShowMessage("Розмірності матриць не узгоджені!!!");}
}
/*-----*/
dmatrix fn_TMatrix0 (dmatrix Q) {
int N = Q.size();
int M = Q[0].size();
dmatrix T;
T.resize(M);
for (int i = 0; i < M; i++) {
T[i].resize(N);
for (int j = 0; j < N; j++)
{ T[i][j] = Q[j][i]; }
}
return T;
}
```

```

/*-----*/
dmatrix fn_IMatrix0 (dmatrix QMatrix) {
int Q = QMatrix.size();
if (QMatrix[0].size()!=Q) {
ShowMessage("У матриці неузгоджені
кількості рядків та стовбців!!!");
}
else {
dmatrix OutMatrix;
Matrix InvM(Q);

for(int i = 0; i < Q; i ++)
{
for(int j = 0; j < Q; j ++)
{ InvM.SetElement(i,j, QMatrix[j][i]); }
}

if (InvM.Determinant()==0)
{ ShowMessage("Неможливо визначити обернуту матрицю!!!"); }
else
{
InvM=InvM.Inverse();
OutMatrix.resize(Q);
for(int i = 0; i < Q; i ++)
{
OutMatrix[i].resize(Q);
for(int j = 0; j < Q; j ++)
{ OutMatrix[i][j] = InvM.GetElement(i, j); }
}
}
return OutMatrix; }
}
/*-----*/

```

Програмний код головного файлу, що реалізує інтерфейс та функціонал додатку, можна написати у такому вигляді

```

//-- Unit01.cpp -----*/
#include <vcl.h>
#pragma hdrstop

#include "Unit01.h"

```

```

#include "Matrix.h"
/*-----*/
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"

TfrmUnit01 *frmUnit01;
typedef TButton* TButtons;

int Q;
double Ka,Kb,Kc, Qa,Qb,Qc;
TStringGrid *Grid0,*Grid1,*GridT, *Matrix_Q;
TEdit *Edit_Q;
dmatrix QToMem, InvMatrix, M0,M1,MT, A,B,C;
AnsiString MatrixName;
AnsiString Caption0 = "Розрахована матриця ";

dmatrix SetMatrix0 (TStringGrid *Matrix_Q)
{
int I = Matrix_Q->RowCount;
int J = Matrix_Q->ColCount;
dmatrix Q;
Q.resize(I);
for(int i = 0; i < I; i ++)
{
Q[i].resize(J);
for(int j = 0; j < J; j ++)
{
Q[i][j] = StrToFloat(Matrix_Q->Cells[j][i]);
}
}
return Q;
}

void PutMatrix (dmatrix X)
{
int I = X.size();
int J = X[0].size();
Matrix_Q->RowCount = I;
Matrix_Q->ColCount = J;
for(int i = 0; i < I; i ++)
{

```

```

for(int j = 0; j < J; j ++)
{
Matrix_Q->Cells[j][i] = FloatToStr(X[i][j]);
}
}
}

```

```

void AlgMatrix (dmatrix X, TStringGrid *Matrix_Q)
{
int I = X.size();
int J = X[0].size();
Matrix_Q->RowCount = I;
Matrix_Q->ColCount = J;
for(int i = 0; i < I; i ++)
{
for(int j = 0; j < J; j ++)
{
Matrix_Q->Cells[j][i] = FloatToStr(X[i][j]);
}
}
}

```

```

void ZeroMatrix (TStringGrid *Matrix_Q, TEdit *Edit_Q)
{
int I = Matrix_Q->RowCount;
int J = Matrix_Q->ColCount;
for(int i = 0; i < I; i ++)
{
for(int j = 0; j < J; j ++)
{
if (Matrix_Q->Cells[j][i] == "") {Matrix_Q->Cells[j][i] = "0";}
}
}
Edit_Q->Text = "1";
}

```

```

void ClearMatrix (TStringGrid *Matrix_Q, TEdit *Edit_Q)
{
int I = Matrix_Q->RowCount;
int J = Matrix_Q->ColCount;
for(int i = 0; i < I; i ++)

```

```

{
for(int j = 0; j < J; j ++)
{
Matrix_Q->Cells[j][i] = "0";
}
}
Edit_Q->Text = "1";
}

/*-----*/
__fastcall TfrmUnit01::TfrmUnit01(TComponent* Owner) : TForm(Owner)
{
}
/*-----*/
void __fastcall TfrmUnit01::FormShow(TObject *Sender)
{
frmUnit01->Width = 1516;
Panel0->Width = 15;
Panel1->Width = Panel0->Width;
Panel2->Width = Panel0->Width;
Panel3->Width = Panel0->Width;
PanelA0->Width = 480;
PanelB0->Width = PanelA0->Width;
PanelC0->Width = PanelA0->Width;

TButtons Buttons[12] = {XToMem,Mem,MemToX,Plus0,Minus0,Times0,
Div0,TransX,InvX,PInvX,ClearX,Exit0};
for (int i;i<12;i++) {
Buttons[i]->Width = Footer0->Width/12-3;
Buttons[i]->Left = 15+Buttons[i]->Width*i+2;
}

EditR0Change((TObject *)Sender);

ZeroMatrix (Matrix_A, EditK0);
ZeroMatrix (Matrix_B, EditK1);
ZeroMatrix (Matrix_C, EditK2);
}
/*-----*/
void __fastcall TfrmUnit01::EditR0Change(TObject *Sender)
{

```

```

Q = 0;
if ((EditR0->Text!="")&&(EditC0->Text!="")) {
int N = StrToInt(EditR0->Text);
int M = StrToInt(EditC0->Text);
Matrix_A->RowCount = N;
Matrix_A->ColCount = M;
EditR1->Text = M;
if (N==M) {Q = N;}
Matrix_C->RowCount = N;
Matrix_C->ColCount = M;
ZeroMatrix (Matrix_A, EditK0);
ZeroMatrix (Matrix_C, EditK2);
}}
/*-----*/
void __fastcall TfrmUnit01::EditR1Change(TObject *Sender)
{
Q = 0;
if ((EditR1->Text!="")&&(EditC1->Text!="")) {
int N = StrToInt(EditR1->Text);
int M = StrToInt(EditC1->Text);
Matrix_B->RowCount = N;
Matrix_B->ColCount = M;
EditC0->Text = N;
if (N==M) {Q = N;}
Matrix_C->RowCount = N;
Matrix_C->ColCount = M;
ZeroMatrix (Matrix_B, EditK1);
ZeroMatrix (Matrix_C, EditK2);
}}
/*-----*/
void __fastcall TfrmUnit01::InvXClick(TObject *Sender)
{
Matrix InvM(Q);

for(int i = 0; i < Q; i ++)
{
for(int j = 0; j < Q; j ++)
{
InvM.SetElement(i, j, StrToFloat(Matrix_Q->Cells[i][j]));
}
}
}

```

```

if((InvM.Determinant()==0) || (Q==0))
{
ShowMessage("Неможливо визначити обернуту матрицю!!!");
ClearMatrix (Matrix_C, EditK2);
}
else
{
InvM=InvM.Inverse();
LabelC0->Caption = Caption0+InvX->Caption;

InvMatrix.resize(Q);
for(int i = 0; i < Q; i ++)
{
InvMatrix[i].resize(Q);
for(int j = 0; j < Q; j ++)
{
InvMatrix[i][j] = InvM.GetElement(i, j);
Matrix_C->Cells[i][j] = FloatToStr(InvMatrix[i][j]);
}
}
}
}
else {
ShowMessage("Неможливо визначити обернуту матрицю!!!");
ClearMatrix (Matrix_C, EditK2);
}}
/*-----*/
void __fastcall TfrmUnit01::Matrix_AEnter(TObject *Sender)
{
Matrix_Q = Matrix_A;
MatrixName = "A";
Edit_Q = EditK0;
EditK0Change((TObject *)Sender);
if (Matrix_C->Tag!=2) {Matrix_C->Tag=9;
Grid0=Matrix_A; Grid1=Matrix_B; Qa=Ka; Qb=Kb;}
else {Matrix_C->Tag=2;
Grid0=Matrix_C; Grid1=Matrix_A; Qa=Kc; Qb=Ka;}
XToMem->Caption = MatrixName+" to Mem";
MemToX->Caption = "Mem to "+MatrixName;
Plus0->Caption = Grid0->Hint+" Plus "+Grid1->Hint;

```

```

Minus0->Caption = Grid0->Hint+" Minus "+Grid1->Hint;
Times0->Caption = Grid0->Hint+" Times "+Grid1->Hint;
Div0->Caption = Grid0->Hint+" Divide "+Grid1->Hint;
    TransX->Caption = "Trans"+MatrixName;
InvX->Caption = "Inv"+MatrixName;
    PInvX->Caption = "PInv"+MatrixName;
ClearX->Caption = "Clear"+MatrixName;
}
/*-----*/
void __fastcall TfrmUnit01::Matrix_BEnter(TObject *Sender)
{
    Matrix_Q = Matrix_B;
    MatrixName = "B";
    Edit_Q = EditK1;
    EditK0Change((TObject *)Sender);
    if (Matrix_C->Tag!=2) {Matrix_C->Tag=9;
    Grid0=Matrix_B; Grid1=Matrix_A; Qa=Kb; Qb=Ka;}
    else {Matrix_C->Tag=2;
    Grid0=Matrix_C; Grid1=Matrix_B; Qa=Kc; Qb=Kb;}
    XToMem->Caption = MatrixName+" to Mem";
    MemToX->Caption = "Mem to "+MatrixName;
    Plus0->Caption = Grid0->Hint+" Plus "+Grid1->Hint;
    Minus0->Caption = Grid0->Hint+" Minus "+Grid1->Hint;
    Times0->Caption = Grid0->Hint+" Times "+Grid1->Hint;
    Div0->Caption = Grid0->Hint+" Divide "+Grid1->Hint;
        TransX->Caption = "Trans"+MatrixName;
    InvX->Caption = "Inv"+MatrixName;
        PInvX->Caption = "PInv"+MatrixName;
    ClearX->Caption = "Clear"+MatrixName;
}
/*-----*/
void __fastcall TfrmUnit01::Matrix_CMouseUp(TObject *Sender,
    TMouseButton Button, TShiftState Shift, int X, int Y)
{
    if(Button == mbLeft) {
    Matrix_Q = Matrix_C;
    MatrixName = "C";
    Edit_Q = EditK2;
    EditK0Change((TObject *)Sender);
    Matrix_C->Tag = (Matrix_C->Tag!=2) ? 2 : 9;
    XToMem->Caption = MatrixName+" to Mem";

```

```

MemToX->Caption = "Mem to "+MatrixName;
    TransX->Caption = "Trans"+MatrixName;
    InvX->Caption = "Inv"+MatrixName;
    PInvX->Caption = "PInv"+MatrixName;
    ClearX->Caption = "Clear"+MatrixName;
if (Matrix_C->Tag==2) {
Plus0->Caption = Matrix_C->Hint+" Plus ?";
Minus0->Caption = Matrix_C->Hint+" Minus ?";
Times0->Caption = Matrix_C->Hint+" Times ?";
Div0->Caption = Matrix_C->Hint+" Divide ?";
}
else {
Plus0->Caption = "Plus";
Minus0->Caption = "Minus";
Times0->Caption = "Times";
Div0->Caption = "Divide";
}

} else
if(Button == mbRight) {
if (Matrix_C->Tag==2) {
GridT = Grid0;
Qc = Qa;
Grid0 = Grid1;
Grid1 = GridT;
Qa = Qb;
Qb = Qc;
Plus0->Caption = Grid0->Hint+" Plus "+Grid1->Hint;
Minus0->Caption = Grid0->Hint+" Minus "+Grid1->Hint;
Times0->Caption = Grid0->Hint+" Times "+Grid1->Hint;
Div0->Caption = Grid0->Hint+" Divide "+Grid1->Hint;
}
}
}
/*-----*/
void __fastcall TfrmUnit01::XToMemClick(TObject *Sender)
{
int I = Matrix_Q->RowCount;
int J = Matrix_Q->ColCount;
QToMem.resize(I);
QToMem[0].resize(J);

```

```

QToMem = SetMatrix0 (Matrix_Q);
Mem->Caption = MatrixName+" in Mem";
}
/*-----*/
void __fastcall TfrmUnit01::MemClick(TObject *Sender)
{
PutMatrix (QToMem);
    int I = QToMem.size();
    int J = QToMem[0].size();
    Matrix_Q->RowCount = I;
    Matrix_Q->ColCount = J;
    if (MatrixName=="A") {
        EditR0->Text = IntToStr(I);
        EditC0->Text = IntToStr(J);
        EditC0->Anchors = EditR1->Anchors;}
    else
        if (MatrixName=="B") {
            EditR1->Text = IntToStr(I);
            EditC1->Text = IntToStr(J);
            EditR1->Anchors = EditC0->Anchors;}
    else
        if (MatrixName=="C") {}
}
/*-----*/
void __fastcall TfrmUnit01::ClearXClick(TObject *Sender)
{
ClearMatrix (Matrix_Q, Edit_Q);
}
/*-----*/
void __fastcall TfrmUnit01::Exit0Click(TObject *Sender)
{
frmUnit01->Close();
}
/*-----*/

void __fastcall TfrmUnit01::EditK0Change(TObject *Sender)
{
Ka = StrToFloat(EditK0->Text);
Kb = StrToFloat(EditK1->Text);
Kc = StrToFloat(EditK2->Text);
}

```

```

/*-----*/
void __fastcall TfrmUnit01::Plus0Click(TObject *Sender)
{
    TButton *ptButton;
    ptButton=(TButton*)Sender;
    A = SetMatrix0 (Matrix_A);
    B = SetMatrix0 (Matrix_B);
    AnsiString Caption1;
    switch (ptButton->Tag)
    {
    case 3:
        C = Qa*SetMatrix0(Grid0)+Qb*SetMatrix0(Grid1);
        Caption1 = "C=A+B";
        break;
    case 4:
        C = Qa*SetMatrix0(Grid0)-Qb*SetMatrix0(Grid1);
        Caption1 = "C=A-B";
        break;
    case 5:
        C = (Qa*SetMatrix0(Grid0))*(Qb*SetMatrix0(Grid1));
        Caption1 = "C=A*B";
        break;
    case 6:
        InvXClick((TObject *)Sender);
        C = (Qb*InvMatrix)*(Qa*SetMatrix0(Grid0));
        Caption1 = "C=A/B";
        break;
    }
    AlgMatrix (C, Matrix_C);
    LabelC0->Caption = Caption0+Caption1;
}
/*-----*/
void __fastcall TfrmUnit01::TransXClick(TObject *Sender)
{
    dmatrix XMatrix = SetMatrix0 (Matrix_Q);
    dmatrix TMatrix = fn_TMatrix0 (XMatrix);
    AlgMatrix (TMatrix, Matrix_C);
}
/*-----*/
void __fastcall TfrmUnit01::PInvXClick(TObject *Sender)
{

```

```

dmatrix XM = SetMatrix0 (Matrix_Q);
int N = XM.size();
int M = XM[0].size();
dmatrix TM,IM;
TM = fn_TMatrix0 (XM);
if (N <= M) {
IM = fn_IMatrix0 (XM*TM);
IM = TM*IM;
}
else {
IM = fn_IMatrix0 (TM*XM);
IM = IM*TM;
}
Matrix_C->RowCount = IM.size();
Matrix_C->ColCount = IM[0].size();
ZeroMatrix (Matrix_C, EditK2);
AlgMatrix (IM, Matrix_C);
}
/*-----*/

```

В результаті роботи додатку ми отримаємо можливість виконання основних операцій матричної алгебри, які необхідні для розв'язку широкого класу фізико-технічних задач. Суттєво, що відповідальним за виконання матричних операцій є універсальний клас `Matrix` оформлений як заголовочний файл. Цей файл можна звичайним чином підключати в різні проекти, де виникає потреба у матричних обчисленнях.

8.3. Підготовка до комп'ютерного практикуму

- Розглянути теоретичний матеріал щодо операцій матричної алгебри.
- Ознайомитися з теоретичними відомостями щодо теми комп'ютерного практикуму.
- Ознайомитися з рекомендаціями щодо виконання завдання.

8.4. Порядок виконання завдання

- Увімкніть Windows-комп'ютер та запустіть програмне середовище Embarcadero C++ Builder Community Edition або Borland C++ Builder Enterprise v6.0 в залежності від потужності вашого комп'ютера.

- На меню панелі виберіть File->New->Application; File->Save Project As; У діалоговому вікні збереження файлів проекту послідовно перейменуйте та збережіть у папці за вашим вибором файли Unit1.cpp -> Unit01.cpp, Project1 -> Project08.
- Відкомпілюйте проект натиснувши на клавіатурі одночасно клавіши Ctrl+F9. В папці з файлами проекту повинен з'явитися виконавчий файл Project08.exe.
- Перейменуйте головну форму Form1->frmUnit01 і повторно відкомпілюйте проект.
- Згідно рекомендаціям щодо виконання роботи, аналогічно Рис.8.1, розташуйте на головній формі проекту колекцію компонентів (8.1) та надайте їм відповідне оформлення.
- У головному файлі проекту Unit01.cpp підключіть бібліотеки: `#include <Math.hpp>`, `#include <math.h>` та заголовочний файл `#include "Matrix.h"`.
- За допомогою вкладки Event на панелі Object Inspector приєднайте до актуальних компонентів відповідні їм функції та заповніть їх блоки кодами, які реалізують функціонал додатку. В якості орієнтації при написанні кодів можна використати приклад коду головного файлу проекту (див. Рекомендації щодо ...).
- Відкомпілюйте проект та перевірте працездатність додатку. Якщо компіляція виявила помилки у написанні коду — перевірте його, виправте помилки і знову виконайте процедуру компіляції проекту.
- Перевірте коректність роботи додатку на прикладі обчислення псевдооберненої матриці

$$\hat{A} = \begin{bmatrix} 1 & -1 & 2 & 0 \\ -1 & 2 & -3 & 1 \\ 0 & 1 & -1 & 1 \end{bmatrix}$$

В результаті застосування додатку для обчислення псевдооберненої

матриці повинна бути отримана матриця

$$\hat{A}^{-1} = \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{9} & \frac{1}{9} & \frac{2}{9} \\ \frac{2}{9} & -\frac{1}{9} & \frac{1}{9} \\ \frac{4}{9} & \frac{1}{9} & \frac{5}{9} \end{bmatrix}$$

8.5. Зміст звіту щодо виконаної роботи

При допуску до виконання завдання проект звіту повинен містити:

- мету комп'ютерного практикуму;
- короткі теоретичні відомості.

При захисті виконаної роботи повинно бути:

- надано відкомпільований виконавчий файл проекту;
- до звіту включено:
 - тексти заголовочного файлу **Matrix.h** та головного файлу **Unit01.cpp**;
 - результати тестування додатку;
 - аналіз отриманих результатів та висновки.

8.6. Контрольні запитання

- В чому заключається головна відмінність об'єктно-орієнтованого програмування від структурного програмування?
- Що таке об'єкт класу і яким чином його використовують в програмуванні?
- Чому при програмуванні додатку для виконання матричних операцій доцільно використати клас **Matrix**?
- Що таке перезавантаження матричних операцій?
- Чим відрізняється обернена матриця від псевдооберненої матриці і у яких випадках вони співпадають?

9. Список використаної літератури

1. **Основи алгоритмізації та програмування: Навчальний посібник**, *М.М.Чепілко*, — Київ: КПІ ім. Ігоря Сікорського, 2022. — 139 с.
2. **Вступ до програмування мовою C++. Організація обчислень: Навчальний посібник**, *Ю.А.Бєлов, Т.О.Карнаух, Ю.В.Коваль, А.Б.Ставровський*, — Київ: Видавничо-поліграфічний центр "Київський університет", 2012. — 175 с.
3. **Алгоритмізація та програмування: Практикум. Навчальний посібник**, *Л.І.Кублій*, — Київ: КПІ ім. Ігоря Сікорського, 2019. — 209 с.
4. **Уроки програмування на C++**, *Переклад з Ravesli / Уроки по C++* — WEB-ресурс, 2022.
5. **Embarcadero C++ Builder Community Edition** — Інтернет-ресурс
6. **Borland C++ Builder Enterprise v6.0** — Інтернет-ресурс
7. **Загальний курс фізики. Механіка, молекулярна фізика і термодинаміка**. *Кучерук І.М., Горбачук І.І., Луцик П.П.* — К: Техніка, 1999.
8. **Загальний курс фізики. Електрика і магнетизм**. *Кучерук І.М., Горбачук І.І., Луцик П.П.* — К: Техніка, 2001