Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

Іліка С.А., Перцов А.С., Юрченко І.В.

ОБЧИСЛЮВАЛЬНА ПРАКТИКА

методичні вказівки до обчислювальної практики
для студентів другого курсу спеціальностей
6.122 – Комп'ютерні науки, 6.124 – Системний аналіз

Чернівці – 2022

ББК 32.973.2-018
УДК 004.021

Друкується за ухвалою редакційно-видавничої ради
Чернівецького національного університету
імені Юрія Федьковича

Обчислювальна практика. Методичні вказівки до обчислювальної практики для студентів другого курсу спеціальностей 6.122 – Комп’ютерні науки, 6.124 – Системний аналіз / Укл.: Іліка С. А., Перцов А.С., Юрченко І.В.– Чернівці: Чернівецький національний університет імені Юрія Федьковича, 2022. – 35 35 с.

Обчислювальна практика студентів є складовою частиною навчального процесу і проводиться з метою закріплення й поглиблення теоретичних знань, набуття навичок і досвіду самостійної практичної роботи на комп'ютері.

Завдання обчислювальної практики призначені допомогти студентам закріпити й поглибити знання з навчальних дисциплін циклу програмування (“Об’єктно-орієнтоване програмування”, “Бібліотеки мови Python”, “Обчислювальна геометрія та комп’ютерна графіка”, “Пакети прикладних програм”).

Для студентів другого курсу спеціальностей 6.122 – Комп’ютерні науки, 6.124 – Системний аналіз.

ББК 32.973.2-018
УДК 004.021

ЗМІСТ

1. Вступ

Питання проведення практик в Чернівецькому національному університеті імені Юрія Федьковича регулюється “Положенням про проведення практики здобувачів вищої освіти Чернівецького національного університету імені Юрія Федьковича” (затверджено на засіданні Вченої ради ЧНУ, протокол №7 від 31 серпня 2020 року) [1].

Дані методичні вказівки до обчислювальної практики призначені для студентів другого курсу Чернівецького національного університету імені Юрія Федьковича (ЧНУ), що навчаються за спеціальностями 122 – Комп’ютерні науки, 124 – Системний аналіз (галузь знань – 12 – Інформаційні технології).

Методичні вказівки є допоміжним матеріалом для керівника практики і студентів, містять початкові відомості про порядок організації, проведення і завершення практики відповідно до рекомендацій Вченої ради ЧНУ та зразки завдань з практики.

2. Цілі і завдання обчислювальної практики

Обчислювальна практика має на меті закріплення знань, отриманих в процесі вивчення дисциплін з програмування, а також отримання навичок розв’язання задач шляхом програмування на мовах високого рівня.

За час обчислювальної практики студент повинен виконати практичне завдання, що включає основні етапи від постановки завдання до отримання остаточного результату: алгоритмізація, програмування, розв’язання задачі на комп'ютері та оформлення звіту.

3. Положення про обчислювальну практику

3.1 Загальні положення

3.1.1. Обчислювальна практика студентів очної форми навчання проводиться після завершення екзаменаційної сесії четвертого навчального семестру. Тривалість практики 2 тижні.

3.1.2. Обчислювальна практика проводиться в комп’ютерних класах факультету або, з урахуванням заходів протиепідемічного характеру, дистанційно.

3.2 Методичне і організаційне керівництво

3.2.1 Для методичного і організаційного керівництва практикою призначаються керівники практики від кафедр.

3.2.2 Керівник практики забезпечує проведення наступних організаційних заходів:

- бере участь в підготовці методичних матеріалів по практиці, надає студентам консультативну допомогу з питань організації практики;
- організовує і контролює проведення практики відповідно до програми і графіку проходження практики;
- організовує проведення консультацій (за необхідності);
- контролює дотримання студентами виробничої дисципліни;
- здійснює постійний контроль за роботою студентів-практикантів, допомагає їм правильно виконувати завдання на робочому місці;
- контролює підготовку звітів, дає оцінку практикантові з урахуванням виконання програми практики та індивідуального завдання.

3.3 Обов'язки студента на практиці

У період проходження обчислювальної практики студент зобов'язаний:

- виконувати завдання, передбачені програмою та індивідуальним завданням студента на практику;
- підготувати і представити звіт керівникові практики.

3.4 Підведення підсумків практики

3.4.1. Після закінчення практики студент складає звіт. Звіт повинен містити відомості про виконану в період практики роботу.

3.4.2. Завдання, код програми, результати виконання студенти повинні оформити у вигляді звіту та завантажити його для перевірки на сторінку обчислювальної практики на сайті дистанційного навчання https://moodle.chnu.edu.ua.

3.4.3. Файли з текстом програми, що реалізовує розрахунки за пунктами завдання, додаються до звіту в електронному вигляді. Програма має бути повністю працездатна.

3.4.4. Студент, що не виконав програму практики, отримав негативну оцінку при захисті звіту або незадовільний відгук про роботу, спрямовується на практику повторно. В окремих випадках ректор розглядає питання про перебування студента у ВНЗ.

4. Програма практики

4.1. Зміст обчислювальної практики

Програма практики включає наступні пункти:

- знайомство з методами розв’язування запропонованої задачі;

- розробка і написання програми для розв’язування індивідуального завдання на запропонованій мові програмування високого рівня;
- засвоєння принципів організації інтерфейсу (меню, діалогові вікна);
- оформлення звіту з практики.

4.2 Перелік тем індивідуальних завдань

Відповідно до завдань практики студент виконує індивідуальне завдання. Програмна частина завдання має бути виконана на мовах високого рівня. Завдання вибираються із запропонованого блоку завдань, яке узгоджується з керівником практики.

При оцінці роботи велике значення надається якості оформлення звіту, який має бути виконаний з безумовним дотриманням вимог п. 5.1 методичних вказівок. Беруться до уваги якість програмного рішення і оформлення тексту програми (зручність читання, наявність коментарів і тому подібне). Максимальну оцінку за практику студент отримує лише в разі повного розв’язання задачі і правильного оформлення звіту.

5. Оформлення звіту та атестація

5.1 Оформлення звіту з практики

Звіт повинен містити всі необхідні пояснювальні та розрахункові матеріали та мати наступну структуру:

1. Титульний аркуш.
2. Завдання на практику.
3. Зміст.
4. Основна частина.
5. Висновок.
6. Додатки.

Приклад оформлення титульного аркуша наведений в Додатку.

Зміст містить найменування всіх розділів, підрозділів і пунктів, список використаної літератури, додатка з вказівкою номерів сторінок, на яких вони починаються.

Основна частина звіту повинна містити:

- теоретичний аналіз поставленого завдання, порівняння різних методів розв’язування задачі (якщо присутні декілька методів розв'язання);
- обґрунтування вибору алгоритму;
- опис алгоритму;
- графічні ілюстрації, що пояснюють роботу програми;
- результати роботи програми.

У додаток виносяться:

- код програми з коментарями, для кожної підпрограми повинно бути вказано, що вона робить, що є вхідними даними і результатом;
- результати тестування програми, тобто розв'язування обчислювальної задачі з різними початковими умовами і заздалегідь відомим результатом.

5.2 Атестація студентів за результатами практики

5.2.1 До атестації допускаються студенти, що представили звіт з обчислювальної практики.

5.2.2 Перевірку звіту здійснює керівник практики.

6. Завдання обчислювальної практики

Приклад роботи з графікою у C++

Інкапсульовані функції GDI і WinApi об'єктного класу канви відносять до трьох різних рівнів. У цій класифікації функції високого рівня забезпечують можливість малювання ліній, фігур і тексту. Визначення властивостей і методів маніпулювання графічними примітивами канви віднесені до середнього рівня. Нижній рівень забезпечується доступ до самих функцій Windows GDI.

Рівень	Метод (Функція)	Властивості	Дія
Високий	MoveTo	PenPos	Визначає поточну позицію пера
	LineTo	PenPos	Малює пряму до заданої точки
	Rectangle		Малює прямокутник
	Ellipse		Малює еліпс
	Arc		Малює дугу
	Polyline		Малює ламану лінію
	PolyBezier		Малює криву Блейзера
	Chord		Малює сектор
	DrawFocusRect		Малює прямокутник
	FrameRect		Виводить рамку навколо прямокутника
	Pie		Виводить сектор круга
	TextOut		Виводить текстовий рядок
	TextHeight		Задає висоту текстового рядка
	TextWidth		Задає ширину для виведення текстового рядка
	TextRect		Виведення тексту всередині прямокутника
	FillRect		Заливка вказаного прямокутника кольором і текстурою поточного пензля
	FloodFill		Заливка області канви (довільної форми) заданим кольором
Середній	Pen		Використовується для встановлення кольору, стилю, ширини і режиму пера
		Brush	Використовується для встановлення кольору і текстури пензля при заливці графічних фігур і фону канви

Рівень	Метод (Функція)	Властивості	Дія
		Font	Використовується для установки шрифту заданого кольору, розміру і стилю шрифта
		Pixels	Використовується для читання і запису кольору заданого пікселя канви
	CopyRect	CopyMode	Копіює прямокутну область канви в режимі CopyMode
	BrushCopy		Копіює прямокутну область канви із заміною кольору
	Draw		Малює бітовий образ, піктограму, метафайл в заданому місці канви
	StretchDraw		Малює бітовий образ, піктограму або метафайл так, щоб цілком заповнити заданий прямокутник
Низький		Handle	Використовується як параметр при виклику функцій Windows GDI

Приклади використання функцій канви для малювання примітивів

Простий приклад відноситься до малювання ліній. Також ілюструється, як саме можна задавати параметри пера:

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 //Задаємо колір пера
 Canvas ->Pen ->Color=(TColor) RGB(255,0,0);
 //Задаємо ширину пера
 Canvas ->Pen ->Width=5;
 //Можна перемістити перо у вихідну точку так
 Canvas ->MoveTo(100,200);
 //Чи перемістити перо так
 TPoint tPoint;
 tPoint.x=100;
 tPoint.y=200;
 Canvas ->PenPos=tPoint;
 //І малюємо лінію від вихідної точки 100,200 до кінцевої 0,50
 Canvas ->LineTo(0,50);
 //Звільняти і відновлювати нічого не потрібно,
 //проте канва запам'ятовує встановлені параметри
}
```

Приклад малювання дуги і секторів

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 //Стиль пензля
 Canvas ->Brush ->Style=bsHorizontal;
 //Колір пензля
 Canvas ->Pen ->Color = clBlue;
```

```
 //Малюємо дугу
 Canvas ->Arc(0,0,500,500,250,0,50,0);
 //Малюємо сектор, змінюючи стиль взаємодії кольору пера і полотна
 Canvas ->Pen ->Mode=pmWhite;
 Canvas ->Chord(0,0,250,250,250,125,0,0);
 //Звільняти і відновлювати нічого не потрібно
 //Але пам'ятаємо, що канва запам'ятала встановлені параметри
}
```

Приклад малювання ламаних ліній

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 TPoint tPoints[6];
 Canvas ->Pen ->Color = clRed;
 Canvas ->Pen ->Width=3;
 tPoints[0].x = 40;
 tPoints[0].y = 10;
 tPoints[1].x = 20;
 tPoints[1].y = 60;
 tPoints[2].x = 70;
 tPoints[2].y = 30;
 tPoints[3].x = 10;
 tPoints[3].y = 30;
 tPoints[4].x = 60;
 tPoints[4].y = 60;
 tPoints[5].x = 40;
 tPoints[5].y = 10;
 Canvas ->Polyline(tPoints, 5);
}
```

Приклад малювання кривих Блейзера

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 TPoint tPoints[7];
 tPoints[0]=TPoint(0,0);
 tPoints[1]=TPoint(800,30);
 tPoints[2]=TPoint(0,40);
 tPoints[3]=TPoint(550,400);
 tPoints[4]=TPoint(350,200);
 tPoints[5]=TPoint(550,400);
 tPoints[6]=TPoint(0,500);
 Canvas ->PolyBezier(tPoints, 6);
}
```

Приклад відображення прямокутників і еліпсів різними способами і використання пензлів

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 //Задаємо колір пера
 Canvas ->Pen ->Color=(TColor) RGB(0,255,0);
 //Задаємо ширину пера
 Canvas ->Pen ->Width=1;
 //Стиль пера - пунктир
 Canvas ->Pen ->Style=psDot;
 //Стиль виведення замкненої фігури, залежний від кольору пера
 //і канви (замінює своїми можливостями) функцію SetBkMode()
 // прозорий фон
 Canvas ->Pen ->Mode=pmCopy;
 //Малюємо прямокутник по точках
 Canvas ->Rectangle(0,0,100,100);
 // аналог GDI непрозорого фону
 Canvas ->Pen ->Mode=pmWhite;
 //Малюємо еліпс, вписаний у прямокутник
 Canvas ->Ellipse(250,250,350,550);
 //Створюємо перо функцією CreatePen() 1 - товщина пера
 HPEN hPen=CreatePen(PS_DASHDOTDOT,1, RGB(255,0,0));
 //Встановлюємо це перо як поточне
 Canvas ->Pen ->Handle=hPen;
 //Змінюємо стиль виведення
 Canvas ->Pen ->Mode=pmCopy;
 //Стиль пензля – вертикальне штрихування
 Canvas ->Brush ->Style=bsVertical;
 //Можна перевизначити прозорість і таким чином
 SetBkMode(Canvas ->Handle, OPAQUE);
 //Малюємо прямокутник із закругленими краями
 Canvas ->RoundRect(100,100,200,200,50,50);
 //Координати можна задати і в такий спосіб
 TRect tRect; //Координати точок
 tRect.Left=100; //Ліва
 tRect.Right=500; //Права
 tRect.Top=250; //Верхня
 tRect.Bottom=450; //Нижня
 //Використовуємо пензель для зафарбовування об'єкту
 Canvas ->Brush ->Color=(TColor) RGB(0,0,255);
 Canvas ->Rectangle(tRect);
 Canvas ->Brush ->Color=(TColor) RGB(0,255,255);
 //Стиль пензля
 Canvas ->Brush ->Style=bsCross;
 //Чи по координатах
 Canvas ->RoundRect(100,300,250,450,50,50);
}
```

Використання кисті для заливки фігур

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 Canvas ->Brush ->Color=(TColor) RGB(0,255,255);
 TRect tRect(0,0,100,100);
 Canvas ->FillRect(tRect);
 //Малюємо прямокутник по точках
 Canvas ->Rectangle(0,0,100,100);
 //Звільняти і відновлювати нічого не потрібно
}
```

Ефект заповнення канви кольором пензля

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 //Параметр fsBorder - заповнити усю область
 //кольором пензля, до краю канви
 Canvas ->Brush ->Color=(TColor) RGB(255,0,255);
 //Вихідна точка в центрі канви, у даному випадку
 //вікна застосування
 Canvas ->FloodFill(Width/2, Height/2, NULL, fsBorder);
 //Міняємо колір і повторюємо
 Canvas ->Brush ->Color=(TColor) RGB(255,0,0);
 Canvas ->FloodFill(Width/2, Height/2, NULL, fsBorder);
}
```

Малювання рамки навколо прямокутника

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 Canvas ->Brush ->Color=(TColor) RGB(125,0,255);
 TRect tRect; //Координати точок
 tRect.Left=100; //Ліва
 tRect.Right=500; //Права
 tRect.Top=250; //Верхня
 tRect.Bottom=450; //Нижня
 Canvas ->FrameRect(tRect);
}
```

Виведення тексту на канву

Для виведення тексту на канву необхідно задати характеристики шрифта (властивість Font канви), текст у форматі AnsiString і використати метод TextOutA() канви.

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

```
{
 AnsiString vasS="Приклад тексту";
 //Колір тексту
 Canvas ->Font ->Color=clRed;
 //Розмір шрифта в точках
 Canvas ->Font ->Size=20;
 //Стиль шрифта
 TFontStyles tFontStyle;
 //Закреслений, похилий, напівгрубий, підкреслений
 tFontStyle << fsStrikeOut << fsItalic << fsBold << fsUnderline;
 Canvas ->Font ->Style =tFontStyle;
 //Ім'я шрифта
 Canvas ->Font ->Name="Times";
 //Виведення тексту
 Canvas ->TextOutA(10,10, vasS);
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
 AnsiString vasS="Приклад тексту";
 Canvas ->Font ->Color=clBlue;
 Canvas ->TextOutA(50,50, vasS);
}
```

Корисні функції для роботи з текстом

TextWidth(AnsiString vasS); – повертає ширину тексту в пікселях, необхідну для відображення рядка vasS заданим шрифтом канви;

TextHeight(AnsiString vasS); – повертає висоту тексту в пікселях, необхідну для відображення рядка vasS заданим шрифтом канви.

У наступному прикладі показана зміна ширини компонента TListBox, при виведенні на його канву тексту, у якого заздалегідь змінений шрифт.

```
void __fastcall
TForm1::Button1Click(TObject *Sender)
{
 //Колір тексту
 ListBox1 ->Canvas ->Font ->Color=(TColor) 0x00FF7D7D;
 //Висота тексту в пікселях
 ListBox1 ->Canvas ->Font ->Height=25;
 AnsiString vasS="Приклад тексту";
 ListBox1 ->Width=ListBox1 ->Canvas ->TextWidth(vasS)+20;
}
void __fastcall
TForm1::Button3Click(TObject *Sender)
{
 AnsiString vasS="Приклад тексту";
 ListBox1 ->Canvas ->TextOutA(2,10, vasS);
}
```

Класи TBitmap, TIcon і TMetafile

Перший приклад показує створення об'єктів відповідних класів, завантаження в них файлів і збереження файлів.

```
//Створюємо об'єкти
Graphics::TBitmap* gBitmap = new Graphics::TBitmap;
Graphics::TIcon* gIcon = new Graphics::TIcon;
Graphics::TMetafile* gMFile = new Graphics::TMetafile;
//Завантажуємо в них файли
gBitmap ->LoadFromFile("1.bmp");
gIcon ->LoadFromFile("1.ico");
//Властивість показує, що файл .emf win32
gMFile ->Enhanced=true;
gMFile ->LoadFromFile("1.emf");
//Збереження зображень у файли
gBitmap ->SaveToFile("1.bmp");
gIcon ->SaveToFile("2.ico");
gMFile ->SaveToFile("2.emf");
delete gBitmap;
delete gIcon;
delete gMFile;
```

Завантаження і малювання графіки

Наступний код використовує клас TBitmap для завантаження зображення з файлу і його малювання, а також для задання растрового зображення пензля. З малюнка, що показує результати роботи програми, видно, що знову для пензля використовується не весь бітовий образ, а тільки перші його 8*8 пікселів.

```
//Створюємо об'єкт TBitmap
Graphics::TBitmap* gBitmap = new Graphics::TBitmap;
//Завантажуємо в нього зображення
gBitmap ->LoadFromFile("1.bmp");
//Малюємо на канві завантажене зображення
Canvas ->Draw(10,10, gBitmap);
//Беремо завантажене зображення в якості пензля,
//з якого використовуються тільки 8*8 пікселів
Canvas ->Brush ->Bitmap = gBitmap;
//Малюємо еліпс
Canvas ->Ellipse(85,10,160,85);
```

Для малювання графіки можна використати функцію StretchDraw(), яка, на відміну від функції Draw() не тільки малює, але й масштабує зображення. Проте іконки не масштабуються, їм тільки виділяється вказаний у функції шматок канви.

```
TRect tRect(10,10,50,50);
```

```
Canvas ->StretchDraw(tRect, gBitmap);
TRect tRect1(100,100,150,150);
Canvas ->StretchDraw(tRect1, gIcon);
TRect tRect2(250,250,350,350);
Canvas ->StretchDraw(tRect2, gMFile);
```

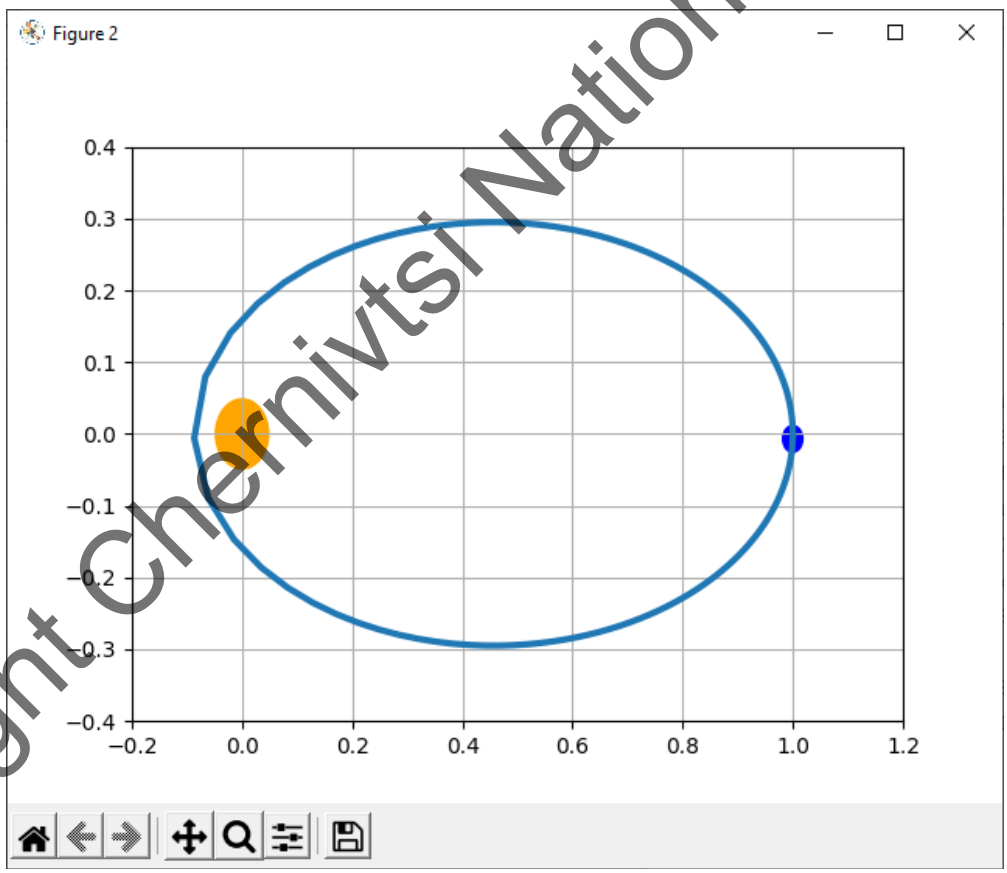
Приклад роботи з графікою у Python

Розробити програму мовою Python з використанням бібліотеки matplotlib для зображення руху небесного тіла навколо Сонця (див. математичну постановку задачі у [7, с.251–253]).

Анімація створюється за допомогою функції FuncAnimation бібліотеки matplotlib.animation у графічному вікні fig2 шляхом багаторазового виклику функції redraw. Значення опції frames=126 підібрано експериментально.

```
import matplotlib.pyplot as plt
from matplotlib.patches import Circle

def f(y, t):
    y1, y2, y3, y4 = y
    return [y2, -y1/(y1**2+y3**2)**(3/2),
            y4, -y3/(y1**2+y3**2)**(3/2)]

t = np.linspace(0,20,1001)
y0 = [1, 0, 0, 0.4]
[y1,y2, y3, y4] = odeint(f, y0, t, full_output=False).T
fig, ax = plt.subplots()
fig.set_facecolor('white')
ax.plot(y1,y3,linewidth=1)
circle = Circle((0, 0), 0.03, facecolor='orange') # круг
ax.add_patch(circle)
plt.axis('equal')
plt.grid(True)

import matplotlib.animation as animation
fig2 = plt.figure(facecolor='white')
ax = plt.axes(xlim=(-0.2, 1.2), ylim=(-0.4, 0.4) )
line, = ax.plot([ ], [ ], lw=3)
circle = Circle((0, 0), 0.05, facecolor='orange')
ax.add_patch(circle)
```

```
pc = plt.Circle((y0[0], y0[2]), 0.02, fc='b')
ax.add_patch(pc)
ax.grid(True)

def redraw(i):
    x = y1[0:i+1]
    y = y3[0:i+1]
    line.set_data(x, y)
    pc.center=(x[-1],y[-1])

anim = animation.FuncAnimation(fig2,
                               redraw,
                               frames=126,
                               interval=50)
plt.show()
```



Завдання № 1. Робота із графікою на мові програмування високого рівня (C, C++, Python та ін.)

Побудувати додаток для виведення графічного зображення геометричної фігури чи кривої (за варіантом). Створюваний додаток має бути з графічним інтерфейсом користувача (GUI):

- вікно додатку,
- робоча область вікна,
- головне меню (вибір параметрів, виконання в майбутньому операцій із завдань 3 та 4) та інші необхідні елементи управління.

Використати механізм об'єктно-орієнтованого програмування при реалізації завдання (для подальшого виконання завдання 3).

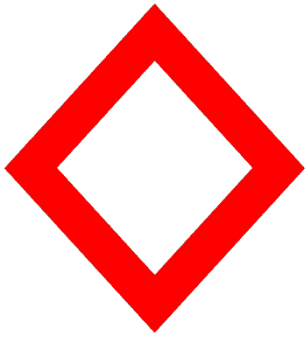
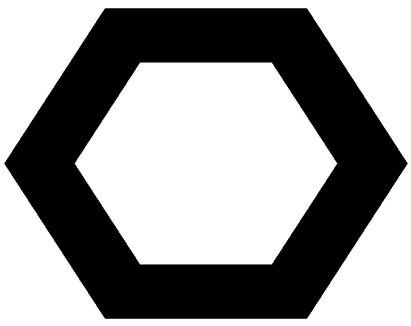
Реалізувати механізм введення параметрів створюваної фігури (кривої).

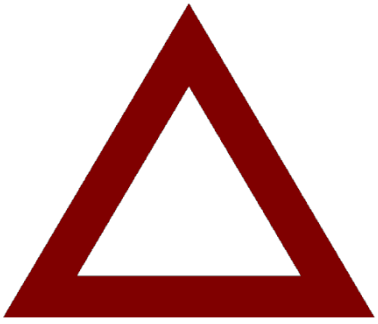
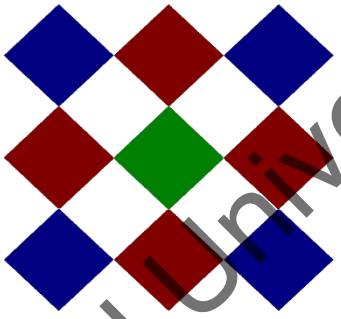
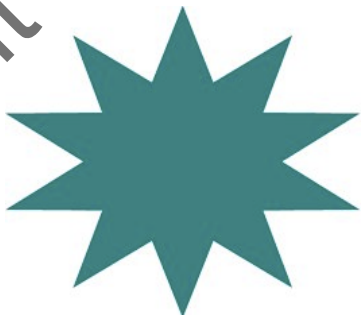
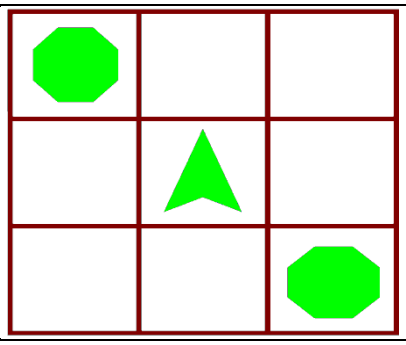
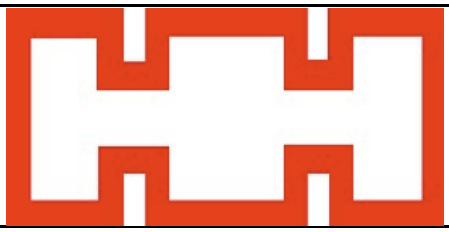
Вибір мови програмування - за бажанням студента (з раніше вивчених при викладанні освітньої програми мов).

Варіанти завдань

1. Побудувати графічне зображення спіралі Архімеда $r=a\varphi$.
2. Побудувати графічне зображення гіперболічної спіралі $r=a/\varphi$ ($r>0$).
3. Побудувати графічне зображення логарифмічної спіралі $r=a\varphi$.
4. Побудувати графічне зображення астроїди.
5. Побудувати графічне зображення кардіоїди.
6. Побудувати графічне зображення чотирипелюсткової троянди $r = a \sin 2\varphi$.
7. Побудувати графічне зображення лемніскати Бернуллі.
8. Побудувати графічне зображення еліпса.
9. Побудувати графічне зображення кривої $r = a \sin 3\varphi$.

Зобразити на екрані малюнок

10.		11.	

12.		13.	
14.		15.	
16.		17.	
18.		19.	
20.		21.	
22.			

Завдання № 2. Робота з математичними пакетами та створення презентації

Самостійно інсталювати пробну (демонстраційну) версію пакету прикладних програм (ППП) та вивчити режими роботи та функціонал пакету. На основі отриманих результатів створити презентацію про роботу з ППП із використанням навігаційних кнопок, гіперпосилань, анімаційних ефектів (Microsoft PowerPoint або інший програмний продукт для створення презентацій, в т.ч. https://prezi.com, https://www.google.com/intl/uk_UA/slides/about/ та ін., не менше 10 слайдів, українською мовою).

Виконати задачі 1 та 2 з використанням одного з ППП.

Виконувати завдання 2 можна за допомогою таких ППП (за вибором студента):

- MathCAD,
- Matlab,
- Maple,
- Wolfram Mathematica,
- бібліотек Numpy, Matplotlib, SciPy, SymPy мови Python.

Теми для презентацій

Maple

1. Математичний аналіз в Maple:
   - границі, суми, ряди;
   - дослідження, розклад, наближення функцій;
   - диференціювання та інтегрування.
2. Розв’язування алгебраїчних рівнянь і нерівностей.
   - команда solve;
   - команда fsolve;
   - розв’язування нерівностей;
   - команди isolve та msolve;
   - різницеві рівняння.
3. Звичайні диференціальні рівняння:
   - аналітичні розв’язки ЗДР;
   - наближені розв’язки ЗДР;
   - числові розв’язки ЗДР;
   - структура DESol;
   - пакет Detools.
4. Двовимірна графіка:
   - структура двовимірної графіки;

- двовимірні команди пакета plottools;
- управляючі параметри двовимірної графіки;
- команда plot;
- спеціальні команди.

Matlab

5. Матричні обчислення:
   - операції над матрицями;
   - лінійна алгебра;
   - розв'язування систем лінійних рівнянь;
   - робота з розрідженими матрицями.
6. Числовий аналіз:
   - робота з поліномами;
   - розв'язування рівнянь та мінімізація;
   - числове інтегрування та диференціювання;
   - розв'язування крайових задач.

MathCad

7. Символьні обчислення:
   - символьна алгебра (спрощення, розклад виразів, поліноми, ряди, розклад на елементарні дроби, підстановка змінної, матрична алгебра);
   - математичний аналіз (диференціювання, інтегрування, розв'язування рівнянь).
8. Числові методи: інтегрування та диференціювання:
   - оператори інтегрування;
   - алгоритми інтегрування;
   - кратні інтеграли;
   - перша похідна;
   - похідні вищих порядків;
   - частинні похідні.
9. Матричні обчислення:
   - найпростіші операції над матрицями (транспонування, додавання, множення, визначники, векторні добутки, символьні операції з матрицями...);
   - матричні функції (створення матриць, злиття та розбиття матриць, норма, ранг, сортування, вивід розмірності...);
   - системи лінійних алгебраїчних рівнянь.
10. Звичайні диференціальні рівняння:
    - ЗДР першого порядку;
    - ЗДР вищих порядків;

- системи ЗДР першого порядку;
- фазовий портрет динамічної системи.

11. Двовимірна графіка:
- XY-графік двох векторів, вектора і ранжованої змінної, функції;
- полярний графік;
- форматування осей, рядів даних;
- зміна розміру і положення графіка...

12. Крайові задачі:
- для ЗДР;
- задачі на власні значення для ЗДР;
- різницеві схеми.

Практичне завдання 1

1. Обчисліть для кожного значення $x = 1, 5, 7$ значення y для наступних функцій:

$$y = \frac{1}{\sqrt[3]{x^2} + \sqrt[6]{x^5}}, \; y = \frac{1}{\sqrt{2 \cdot \pi}} e^{-\frac{x^2}{2}}, \; y = \frac{1}{2}\operatorname{arctg}\left(x^2\right)$$

Рекомендується використати операцію векторизації (MathPalette, матричні операції) Змінить кількість значущих цифр, що виводяться на екран після десяткової крапки, на 6.

2. Розв'язати двома способами (матричним і за допомогою вбудованих функцій) систему лінійних рівнянь :

$$\begin{cases} x - 2y + z = 0 \\ 2x + y - 3z = 1 \\ -x + y + 5z = 1 \end{cases}$$

Змініть нижню межу індексації масивів на 1. Виведіть розв'язок системи рівнянь у вигляді вектора-стовпця та поелементно.

3. Вставте в документ текстову область: "Побудова простого графіку". Визначте функцію $f(x) = \sin(x) + \cos(x) - 1$. Побудуйте графік функції *f*.

4. Задайте вектор V1, що складається з трьох елементів {2,3,1}, і вектор V2 = {4,1,7}. Виконайте наступні операції: V1*3, V1-V2, V1*V2, V1xV2, підсумуйте елементи V1, транспонуйте вектор V2, обчисліть норму вектора V1; використовуючи операцію векторизації, обчисліть sin(V1) та норму одержаного вектора.

5. Задайте матрицю M розмірності 2x3, транспонуйте її.

6. Створіть одиничну матрицю E розмірності 5x5, обчисліть її слід.

7. Створіть дві квадратні матриці M1 і M2 розмірності 3x3, перемножте їх; у отриманої матриці обчисліть визначник, виведіть на екран другий стовпець та поелементно третій рядок.

8. Додайте матриці M1 і M2 (матриця MM), для отриманої матриці обчисліть exp(MM).

9. Об'єднайте матрицю MM і вектор V1, відсортуйте отриману матрицю за першими стовпцем і рядком.

10. Обчисліть власні значення будь-якої з уведених матриць розмірності 3x3, а також власний вектор, що належить другому власному значенню.

11. Задайте ранжовану змінну x, що змінюється в діапазоні від 0 до $\pi/2$ з кроком 0.1; визначте функцію f(x) = x*Sin2(2x), побудуйте її графік. Визначте діапазон зміни цілого індексу i від 0 до 15, $x_i = i/10$, $y_i = x_i \, Sin2(2x_i)$, побудуйте графік функції $y_i(x_i)$.

12. Побудуйте графік функції $g(x,y) = x^2 - y^2$, де змінні x та y змінюються в діапазоні від - 5 до 5.

13. Зобразіть сферу. Її параметричне представлення має вигляд:

$$R = 8 \quad 0 \le \varphi \le 2\pi \quad 0 \le \theta \le \pi$$

$$x(\varphi,\theta) = R \cdot \cos(\varphi) \cdot \sin(\theta)$$

$$y(\varphi,\theta) = R \cdot \sin(\varphi) \cdot \cos(\theta)$$

$$z(\varphi,\theta) = R \cdot \cos(\theta)$$

Кількість точок N=30.

14. Змініть значення радіусу сфери R(f)=|cos(f)|. Побудуйте анімаційний графік (у MathCAD кількість кадрів дорівнює 20, кількість кадрів за секунду – 3). Прогляньте на медіаплеєрі результуючу анімацію. Перед побудовою анімації не забудьте відключити АВТОМАСШТАБ.

15. Побудуйте графіки функцій, заданих полярно:

$$N = 15 \quad \varphi = 0, \frac{1}{N} \ldots 2\pi$$

$$r(\varphi) = 1 + \sin\left(2\varphi + \frac{3\pi}{2}\right)$$

$$r1(\varphi) = 1 + \frac{\sin(3\varphi + \pi)}{2}$$

16. Зобразіть просторову криву:

$$N = 40 \quad i = 1 \ldots N$$

$$x_i = \cos\left(\frac{3\pi}{N} i\right)$$

$$y_i = \sin\left(\frac{3\pi}{N} i\right)$$

$$s_i = \frac{3}{N} i$$

Збільшіть кількість точок N, повторіть побудову попереднього графіка; поекспериментуйте, змінюючи різні параметри відображення графіка.

Практичне завдання 2

1. Знайдіть розв'язки систем рівнянь. Перевірте аналітично, чи всі розв'язки знайдені.

$$\begin{cases} \frac{2}{x} + 3y = 1 \\ -x^2 + 2\sqrt{y} = 2 \end{cases} \qquad \begin{cases} e^{\frac{x}{2}} - y = 0 \\ x^2 - y + 1 = 0 \end{cases}$$

2. Розв'яжіть на відрізку [0,3] задачу Коші, використовуючи:

a) функцію в MathCAD odersolve (у блоці з Given)

b) функцію в MathCAD rkfixed. Виведіть значення отриманого розв'язку в точці x=3.

$$y' = -\frac{e^x}{e^x + 1} \quad y(0) = 0{,}5$$

3. Розв'яжіть задачу Коші для диференціального рівняння на відрізку [0,1], використовуючи:

a) функцію в MathCAD odersolve (у блоці з Given)

b) функцію в MathCAD rkadapt

$$y''' - y' = 0 \quad y'(0) = 1$$

$$y(0) = 3 \quad y''(0) = 1$$

4. Розв'яжіть систему диференціальних рівнянь на відрізку [0,3]. Виведіть значення шуканих функцій та їхніх похідних у точці з координатою x=1.5

$$x'' = 2x - 3y \quad x(0) = 0 \quad x'(0) = 5$$

$$y'' = x - 2y \quad y(0) = 2 \quad y'(0) = -1$$

5. Розв’яжіть крайову задачу для диференціального рівняння на відрізку [0,8], виведіть значення функції та її похідних в кінцевій точці відрізку.

$$y''' + \frac{1}{2} yy'' = 0 \quad y(0) = y'(0) = 0 \quad y'(8) = 1$$

Обчисліть значення функції тільки в кінцевій точці (використайте спеціально призначені для цього форми в MathCAD функцій bulstoer, rkadapt, stiffb, stiffr)

6. Знайдіть корені многочлена

$$y(x) = 2x^3 + 20x^2 - 2x + 100$$

7. Уведіть матрицю координат точок площини:

x	0.1	5	1.1	0.5	3.4	2.7	1.9	4	4.7
y	10	1.3	5	7	3	4.5	6	2.5	1.5

а) побудуйте лінійну і сплайнову інтерполяцію для цієї множини точок;

б) побудуйте функції лінійної, поліноміальної та узагальненої регресії для цих же точок.

8. За допомогою вбудованих функцій введіть 50 випадкових чисел з відрізку [0,2]. Побудуйте функції згладжування даних (за допомогою різних вбудованих функцій).

Завдання № 3. Застосування об’єктно-орієнтованого програмування: успадкування класів, перевантаження операторів

Теоретичні відомості

При роботі з тривимірними об'єктами часто вимагається здійснювати по відношенню до них різні перетворення: рухати, повертати, стискати, розтягувати, скошувати і так далі. При цьому в більшості випадків вимагається, щоб після застосування цих перетворень зберігалися певні властивості.

Будь-яке афінне перетворення задається матрицею 3x3 з ненульовим визначником і вектором перенесення :

$$\vec{p'} = \mathbf{R}\vec{p} + \vec{t}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

R є матрицею лінійного оператора над простором тривимірних векторів. Вектор T потрібний для здійснення паралельного перенесення: якщо множити (0 0 0) на будь-яку матрицю 3x3, знову отримаємо (0 0 0) – початок системи координат відносно перетворення R є нерухомою точкою. Визначник має бути ненульовим, оскільки якщо визначник матриці R дорівнює нулеві, то весь простір переходить у площину, пряму або точку і тим самим не дотримується взаємна однозначність.

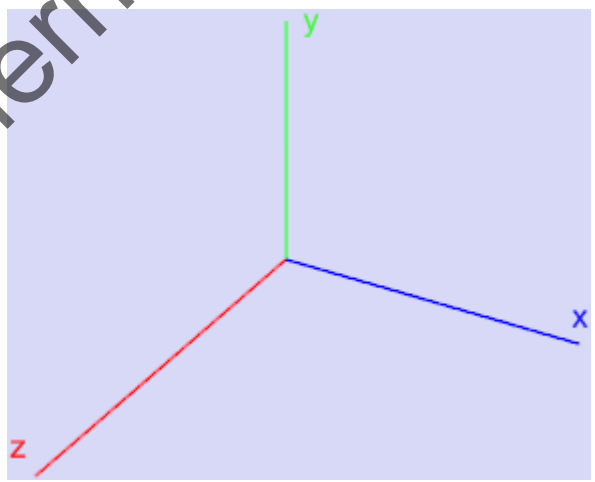
На практиці зручно задавати афінне перетворення однією матрицею. Афінне перетворення задаватиметься наступною матрицею 4x4:

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

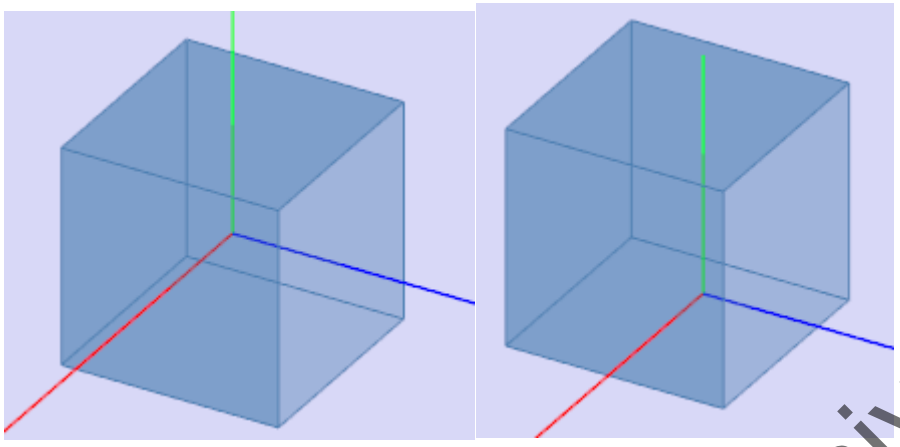
Розглянемо окремі випадки афінних перетворень.

Примітка. Тут і надалі використовуватиметься система координат, уведена таким чином:

- вісь z спрямована на спостерігача, перпендикулярно до площини екрану;
- вісь y знаходиться в площині екрану і спрямована вгору;
- вісь x знаходиться в площині екрану і спрямована праворуч.



Паралельне перенесення



Початковий об'єкт Паралельне перенесення

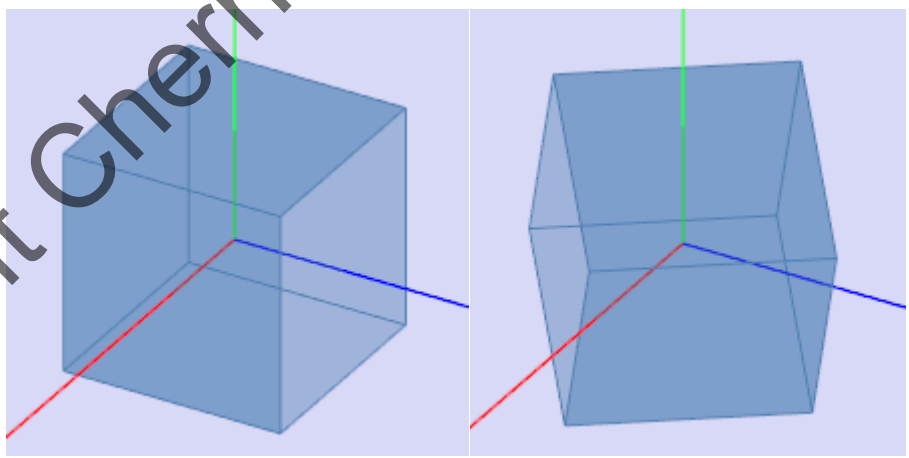
Матриця цього перетворення виглядає таким чином:

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

У даному випадку матриця R = E (одинична матриця).

Перетворення, що розглядаються нижче, зачіпають тільки матрицю R, тому вказуватиметься тільки вона.

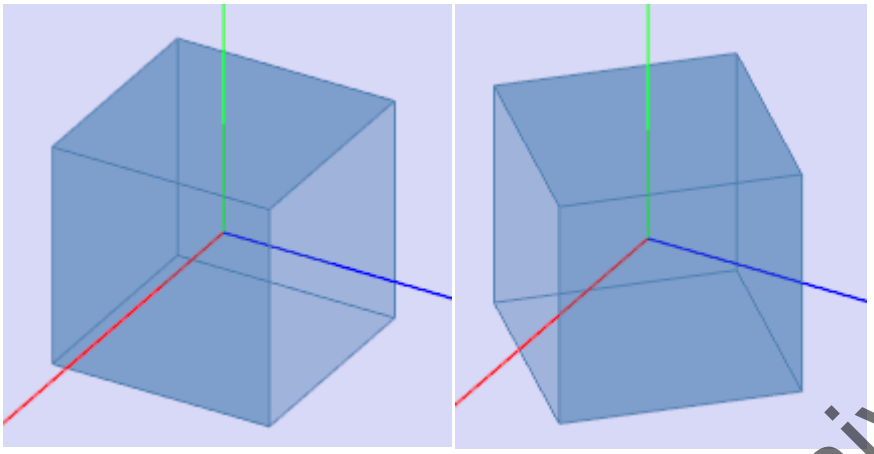
Поворот (обертання)



Початковий об'єкт Поворот навколо деякого вектора

Обертання навколо довільного вектора не тотожне обертанню навколо довільної напрямленої прямої.

Поворот навколо вісі Y



Початковий об'єкт Поворот навколо вісі Y

Зауважимо, що при повороті навколо вісі Y ординати точок (Y-координати) не змінюються. Також варто відмітити, що координати x і z точки перетворяться незалежно від Y-координати. Це означає, що будь-яка точка p(x, y, z) перейде в точку p'(x'(x, z), y, z'(x, y)). Матриця такого перетворення відома:

$$\begin{pmatrix} \cos(-\phi_y) & -\sin(-\phi_y) \\ \sin(-\phi_y) & \cos(-\phi_y) \end{pmatrix}$$

У результаті:

$$x' = x\cos(\phi_y) + z\sin(\phi_y)$$
$$y' = y$$
$$z' = -x\sin(\phi_y) + z\cos(\phi_y)$$

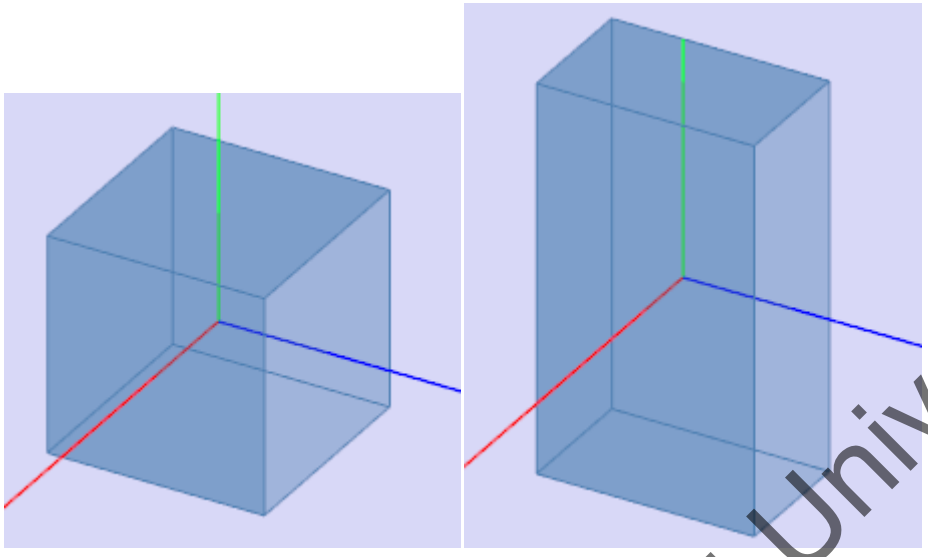
Матриця перетворення Ry(φy) :

$$\begin{pmatrix} \cos(-\phi_y) & 0 & -\sin(-\phi_y) \\ 0 & 1 & 0 \\ \sin(-\phi_y) & 0 & \cos(-\phi_y) \end{pmatrix}$$

Поворот навколо осей x і z :

$$R_x(\phi_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_x) & -\sin(\phi_x) \\ 0 & \sin(\phi_x) & \cos(\phi_x) \end{pmatrix}$$

$$R_z(\phi_z) = \begin{pmatrix} \cos(\phi_z) & -\sin(\phi_z) & 0 \\ \sin(\phi_z) & \cos(\phi_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



Початковий об'єкт Масштабування

Коефіцієнти стискування/розтягування, по аналогії з двомірним простором, визначаються діагональними елементами матриці R :

$$\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$$

У результаті маємо:

$$x' = s_x x$$
$$y' = s_y y$$
$$z' = s_z z$$

Комбінація коефіцієнтів sx = -1, sy = 1, sz = 1 задаватиме відображення від площини Oyz (x=0). При sx = sy = sz = -1 отримаємо центральну симетрію відносно початку координат.

Інтерпретація матриці R

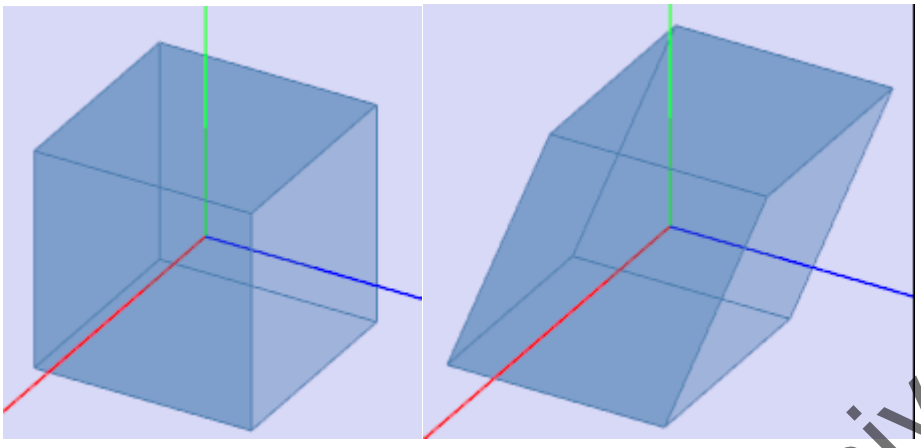
Матриця

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$$

переводить вектору декартового базису:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \to \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$$

Скіс



Початковий об'єкт Скіс

Тепер нескладно отримати перетворення скосу. Наприклад:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Складні афінні перетворення можна отримати як комбінацію простих (елементарних) перетворень. При цьому вибирати прості афінні перетворення можна по-різному.

Завдання

Третє практичне завдання ґрунтується на першій лабораторній роботі і передбачає її попереднє виконання. В ході роботи знадобиться розширити можливості розробленого раніше класу (див. завдання 1), відповідно до вимог індивідуального варіанту.

Виконання завдання містить обов'язкову розробку класу-спадкоємця з реалізованою в ньому додатковою функціональністю. Важливою умовою є те, що передача параметрів об'єктам нового класу повинна здійснюватися за допомогою операторів. Необхідні для кожного варіанту роботи параметри визначені в постановці завдання. Наприклад, для масштабування можна використовувати оператор * («множення»). Інструкція triangle *= 1.5 має виконувати масштабування об'єкту triangle з коефіцієнтом 1.5.

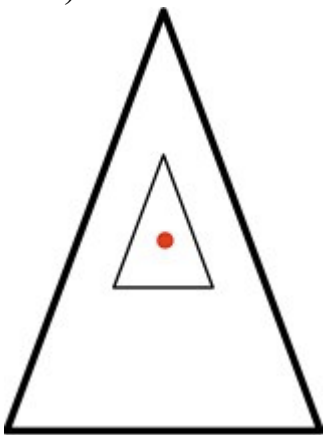
Нижче наведено ілюстрації для різних типів завдань. Тонкими лініями показані початкові фігури, товстими – змінені. Ключові точки виділені.

Студент має виконати всі п’ять запропонованих видів геометричних перетворень.

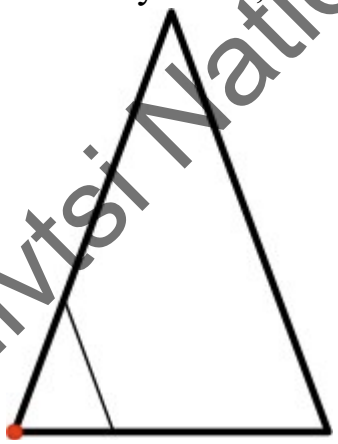
Види перетворень

1. Масштабування фігури відносно геометричного центру. Параметр: площа зміненої фігури.

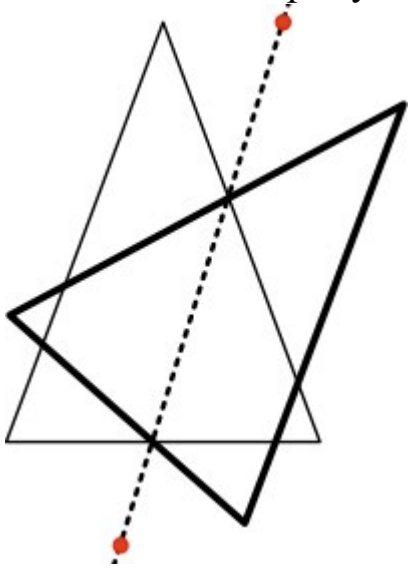
2. Масштабування фігури відносно геометричного центру. Параметр: довжина ребра квадрата, в який повинна вписуватися фігура після зміни (квадрат на площині розташовується так, що його нижнє і верхнє ребро паралельні вісі X).



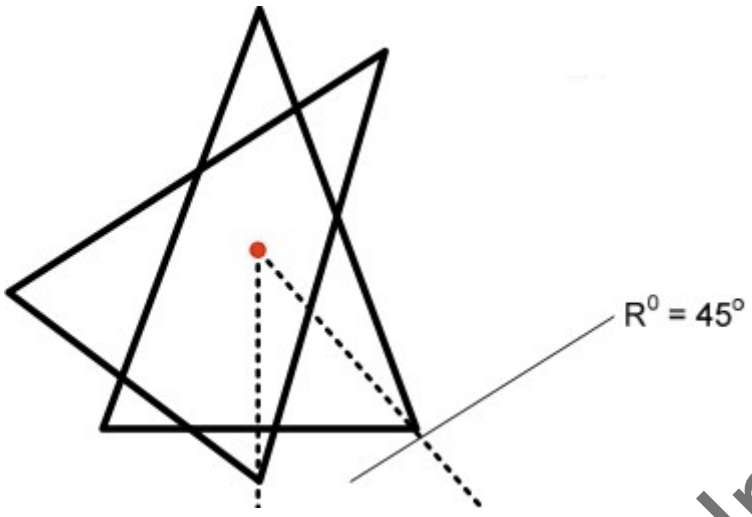
3. Масштабування фігури відносно вибраної точки. Параметри: координати точки центру масштабування, коефіцієнт масштабування.



4. Дзеркальне віддзеркалення фігури відносно прямої. Параметри: координати двох точок, що визначають пряму.



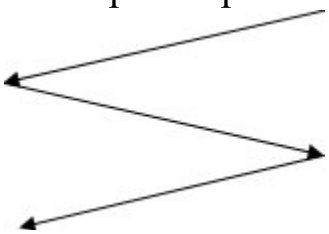
5. Поворот фігури відносно геометричного центру. Параметри: кут повороту.



Завдання № 4. Анімація зображень

Варіанти завдань

1. Зобразити дві хвилі, що рухаються в протилежних напрямках.
2. Зобразити правильний N-кутник і з'єднати кожну вершину з усіма іншими.
3. Зобразити планету, навколо якої рухається супутник.
4. Зобразити коло, що рухається всередині прямокутника, відбиваючись від його сторін.
5. Зобразити коло, яке, починаючи з центру екрана, рухається по спіралі.
6. Зобразити квадрат, який, починаючи з правого верхнього кута екрану, рухається за наступною траєкторією:



7. Зобразити коло, що рухається по синусоїдальній траєкторії, залишаючи за собою слід товщиною в один піксель.
8. Зобразити правильний n-кутник, який рухається по периметру екрану, а в центрі екрану зобразити трикутник, що обертається навколо свого центру.
9. Зобразити три кола, що рухаються навколо центру екрану, не перетинаючись. Перше внутрішнє коло рухається по еліптичній траєкторії, друге – по колу, а третє описує трикутник.

10. Моделювання польоту снаряда. Задається маса снаряда, початкова швидкість, кут нахилу і вітер. В результаті показати траєкторію снаряда, вказати висоту і дальність польоту.

11. Гра “знайди однакові картинки”, що сприяє розвитку зорової пам'яті. Завдання: прибрати всі картки з поля за мінімальну кількість кроків. Під час кроку можна відкрити дві картки, якщо зображення на них однакові, тоді картки зникають, якщо різні, тоді на наступний крок вони закриваються знову.

12. Пасьянс "косинка".

13. Програма "лабіринт". Програма генерує складний лабіринт і розташовує в довільним чином вибрану точку лабіринту фігуру людини. Користувач повинен знайти вихід з лабіринту.

14. Гра "Го".

ЛІТЕРАТУРА ДО ДИСЦИПЛІНИ

1. Положення про проведення практики здобувачів вищої освіти Чернівецького національного університету імені Юрія Федьковича (затверджено на засіданні Вченої ради ЧНУ, протокол №7 від 31 серпня 2020 року) [Електронний ресурс].– Шлях доступу до ресурсу: https://drive.google.com/file/d/1EMTd09rzwmD6gmLzuThArr1uKS6U2Bj6/view
2. Мова програмування С: Методичний посібник та завдання до лабораторних робіт/Укл. Т.М.Сопронюк, І.М. Данилюк.– Чернівці: ЧНУ, 2006.– 72 с.
3. Програмування (інформатика). Методичні вказівки і завдання для обчислювальної практики / Укл.: Строєв О. М., Перцов А.С.– Чернівці: Рута, 2012.– 30 с.
4. Уроки програмування на С++ для початківців [Електронний ресурс].– Шлях доступу до ресурсу: https://acode.com.ua/uroki-po-cpp/
5. С++ Вікіпідручник [Електронний ресурс].– Шлях доступу до ресурсу:https://uk.wikibooks.org/wiki/C%2B%2B
6. Плис А.И., Сливина Н.А. MathCAD: математический практикум для экономистов и инженеров: Учеб. пособие.– М.: Финансы и статистика, 1999.– 656с.
7. Доля П.Г. Введение в научный Python.– Харків: Харківський національний університет ім. В.Н. Каразіна, 2016.– 265 с.
8. Креневич А.П. Python у прикладах і задачах.– Київ: ВПЦ "Київський Університет", 2017. – 206 с.
9. Проценко В.С., Чаленко П.І., Ставровський А.Б. Техніка програмування мовою Сі.– Київ: Либідь, 1993.– 224 с.

Інформаційні ресурси

https://www.maplesoft.com/products/maple/free-trial/?IC=10317

http://mathcad.com.ua/down-math.php

https://uk.mathworks.com/campaigns/products/trials.html?prodcode=ML

https://www.wolfram.com/mathematica/trial/

https://moodle.chnu.edu.ua/course/view.php?id=3667

Додаток 1. Приклад оформлення звіту

Міністерство освіти і науки України
Чернівецький національний університет імені Юрія Федьковича

ЗВІТ
з обчислювальної практики

_____

(прізвище, ім'я, по-батькові)

студента факультету математики та інформатики

Чернівецького національного університету імені Юрія Федьковича

спеціальності _____

курсу_____ групи _____

Чернівці — 202_

Оцінка							
макс.	10	10	10	5	20	25	20
Назва	Завд. 1	пр.з.1	пр.з.2	презент.	Завд. 3	Завд. 4	М-К
		Завдання 2					

Прізвище та ініціали
студента 2__ групи
факультету
математики
та інформатики

Варіант №___

Завдання 1. (умова завдання)

Алгоритм та методи виконання

…….

Завдання 2. (умова завдання)

Алгоритм та методи виконання

…….

Завдання 3. (умова завдання)

Алгоритм та методи виконання

…….

Завдання 4. (умова завдання)

Алгоритм та методи виконання

…….

Навчальне видання

ОБЧИСЛЮВАЛЬНА ПРАКТИКА

методичні вказівки до обчислювальної практики
для студентів другого курсу спеціальностей
6.122 – Комп'ютерні науки, 6.124 – Системний аналіз

Укладачі:
Іліка Світлана Анатоліївна,
Перцов Андрій Сергійович,
Юрченко Ігор Валерійович

Друкується в авторській редакції

Відповідальний за випуск
Черевко І.М.

Підписано до друку 29.06.2022. Формат 60x84/16.

Електронне видання
https://archer.chnu.edu.ua/handle/123456789/24

Чернівецький національний університет імені Юрія Федьковича
58012, Чернівці, вул. Коцюбинського, 2