Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №2

з дисципліни «Моделювання систем в енергетиці»

Тема: «Розробка моделі розрахунку енергетичних потреб на опалення/охолодження будівель»

Виконав:

студент 4-го курсу, НН ІАТЕ

групи ТР-23

Ровний Г. О.

Перевірив:

Волков О.В.

КИЇВ 2025

**Мета роботи:**

Створити комп’ютерну програму розрахунку енергетичних потреб житлових приміщень на опалення/охолодження з використання моделі, яка базується на ДСТУ 9190:2022 «Енергетична ефективність будівель»

**Вхідні дані:**

Огороджуючі поверхні (стіни)

- Кількість стін: 4

- Розміри стін:

1) Стіна № 1: ширина 10 м, висота 2.8 м

2) Стіна № 2: ширина 9 м, висота 2.8 м

3) Стіна № 3: ширина 10 м, висота 2.8 м

4) Стіна № 4: ширина 9 м, висота 2.8 м

**Шари матеріалів (однакові для всіх стін):**

- Штукатурка цементна (зовнішня): товщина 0.02 м; коефіцієнт поглинання 0.4; коефіцієнт випромінювання 0.93

- Цегла силікатна суцільна: товщина 0.25 м

- Теплоізоляція мінеральна вата (базова відсутня): товщина 0.00 м

- Штукатурка внутрішня: товщина 0.015 м

**Додаткові огороджуючі елементи:**

- Перекриття даху (мансардне, холодне горище): металочерепиця, підшив, вентзазор; базова теплоізоляція 0.10 м мінвата

- Підлога над ґрунтом: бетон 0.10 м, стяжка 0.05 м, чорнова підлога; базова теплоізоляція 0.05 м пінополістирол

**Вікна**

- Кількість вікон: 5

- Вікно 1:

- Розміри: 2.0 м × 1.4 м

- Тип скління: подвійне

- Скло: звичайне

- Напрямок: південь

- Вікно 2:

- Розміри: 1.6 м × 1.4 м

- Тип скління: подвійне

- Скло: звичайне

- Напрямок: схід

- Вікно 3:

- Розміри: 1.6 м × 1.4 м

- Тип скління: подвійне

- Скло: звичайне

- Напрямок: захід

- Вікно 4:

- Розміри: 1.2 м × 1.2 м

- Тип скління: подвійне

- Скло: звичайне

- Напрямок: північ

- Вікно 5 (кухня):

- Розміри: 1.4 м × 1.2 м

- Тип скління: подвійне

- Скло: звичайне

- Напрямок: південь

- Двері зовнішні:

- Розміри: 1.0 м × 2.1 м

- Матеріал: метал із теплоізоляційною вставкою

**Інші дані**

**- Площа помешкання: 92 м²**

**- Об’єм приміщення (орієнтовно): 258 м³**

- Температури в приміщенні (цільові):

- Зима: 21°C

- Літо: 25°C

- Місце розташування: **Чернігів**

- Середні температури по місяцях (°C):

- Січень: -5

- Лютий: -4

- Березень: 1

- Квітень: 8

- Травень: 15

- Червень: 19

- Липень: 21

- Серпень: 20

- Вересень: 15

- Жовтень: 8

- Листопад: 2

- Грудень: -3

- Вентиляція (базова): природна інфільтрація, орієнтовний повітрообмін 0.5 крат/год

- Сонячна інсоляція (якісно): фасад південь частково затінений деревами; схід/захід — помірна інсоляція; північ — мінімальна

**Варіанти змін для порівняння**

- Теплоізоляція стін:

- Тип: мінеральна вата або пінополістирол

- Товщина: 0.05–0.15 м (параметр для варіювання)

- Паропроникність/фініш: вітрозахист + фасадна штукатурка

- Заміна вікон:

- Тип скління: енергозберігаюче (Low-E), мультифункціональне

- Заповнювальний газ: повітря / аргон / криптон

- Рамний матеріал: ПВХ із термовставкою або дерев’яні з терморозривом

- Встановлення рекуператора:

- Тип: децентралізований або центральний

- ККД: 70–90% (для моделювання впливу на тепловтрати вентиляцією)

- Зміна температури в приміщенні:

- Параметр: варіювання зимової цільової температури в діапазоні 19–22°C та літньої 24–26°C

**Результат виконання роботи**

**Розрахунок теплових втрат через трансмісію**

Для кожного конструктивного елемента (стіна, вікно, покрівля тощо) теплові втрати необхідно обчислювати за формулою:



U — коефіцієнт теплопередачі елемента (Вт/м²·К),

ΔT — різниця температур між внутрішнім і зовнішнім середовищем

A — площа елемента (м²),

Коефіцієнт теплопередачі U визначаємо за табличними значеннями для матеріалів згідно з ДСТУ:



**Розрахунок теплових втрат через вентиляцію**

Теплові втрати через вентиляцію слід розраховувати за ф-лою:



Об'єм вентиляційного повітря V визначається залежно від кратності повітрообміну в приміщенні.

**Розрахунок теплових надходжень**

Внутрішні теплонадходження обчислюються на основі даних про джерела тепла (освітлення, побутова техніка, люди), а сонячні — залежно від площі вікон, орієнтації приміщення та інтенсивності сонячного випромінювання. Формула для сонячних теплонадходжень:



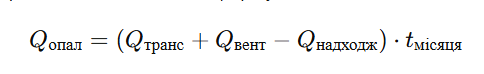
Iсон — інтенсивність сонячного випромінювання (Вт/м²),

Aвікон — площа засклення (м²),

g — коефіцієнт пропускання теплової енергії склом.

**Сумарні теплові потреби на опалення та охолодження**

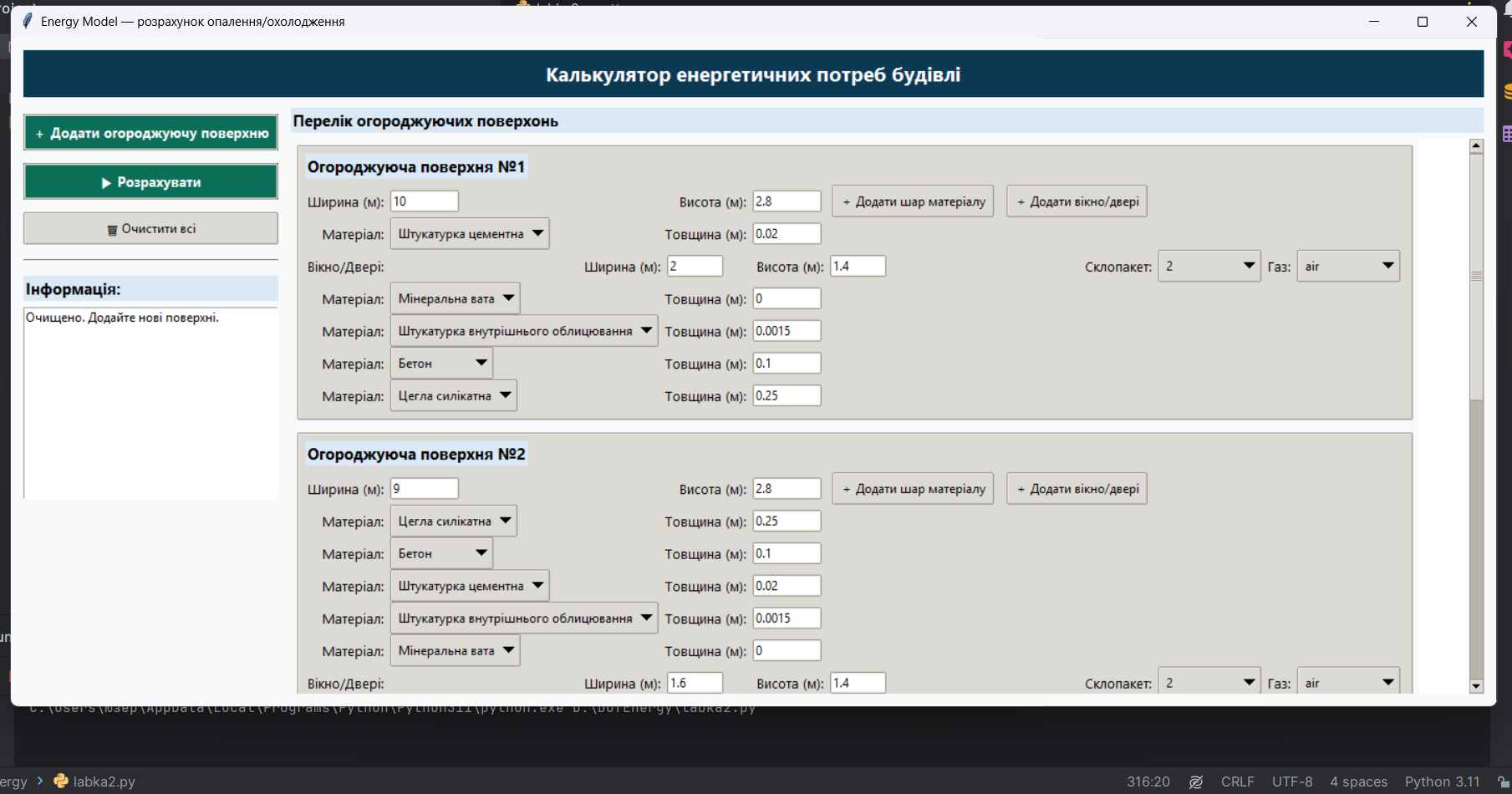
Щомісячні теплові потреби обчислюються за формулою:

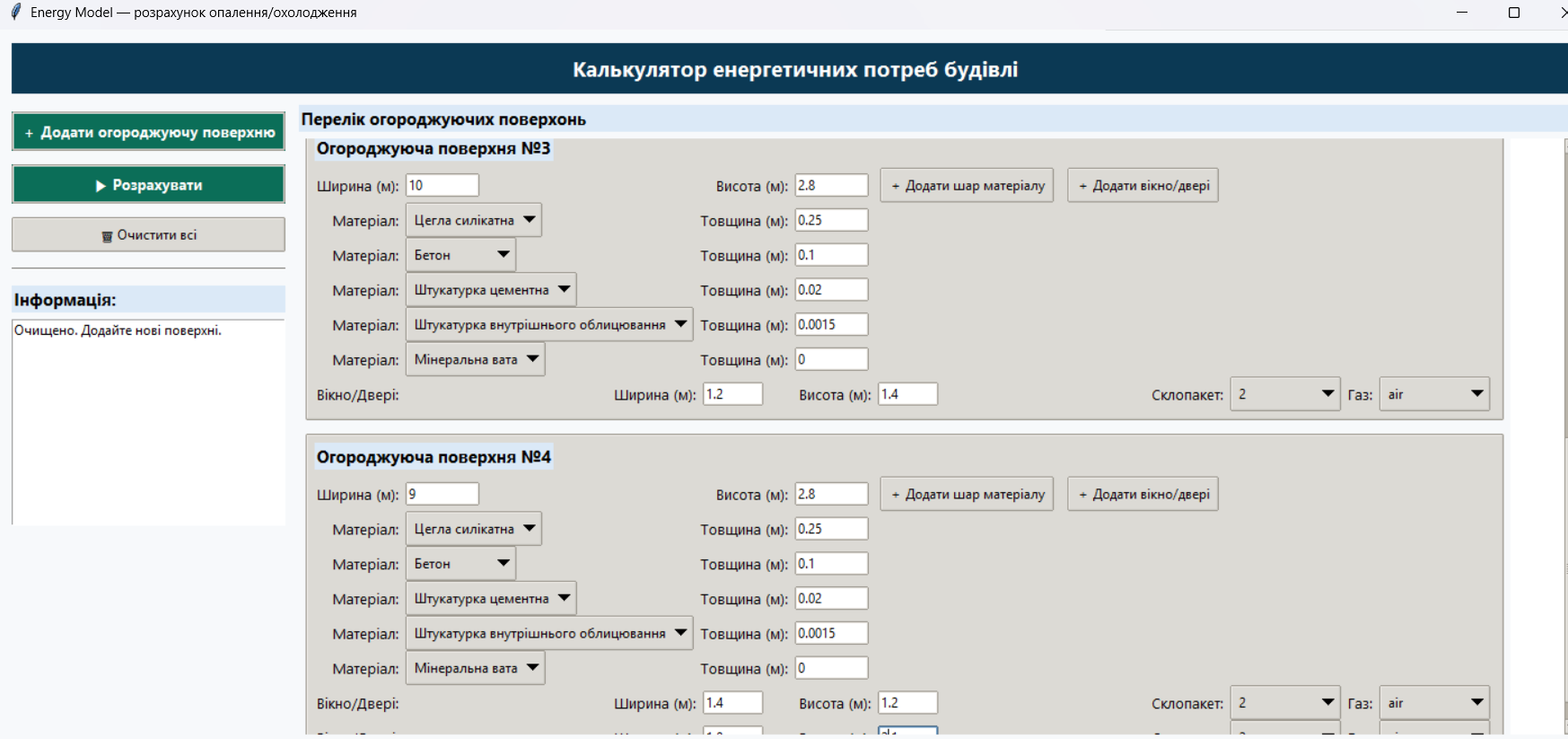


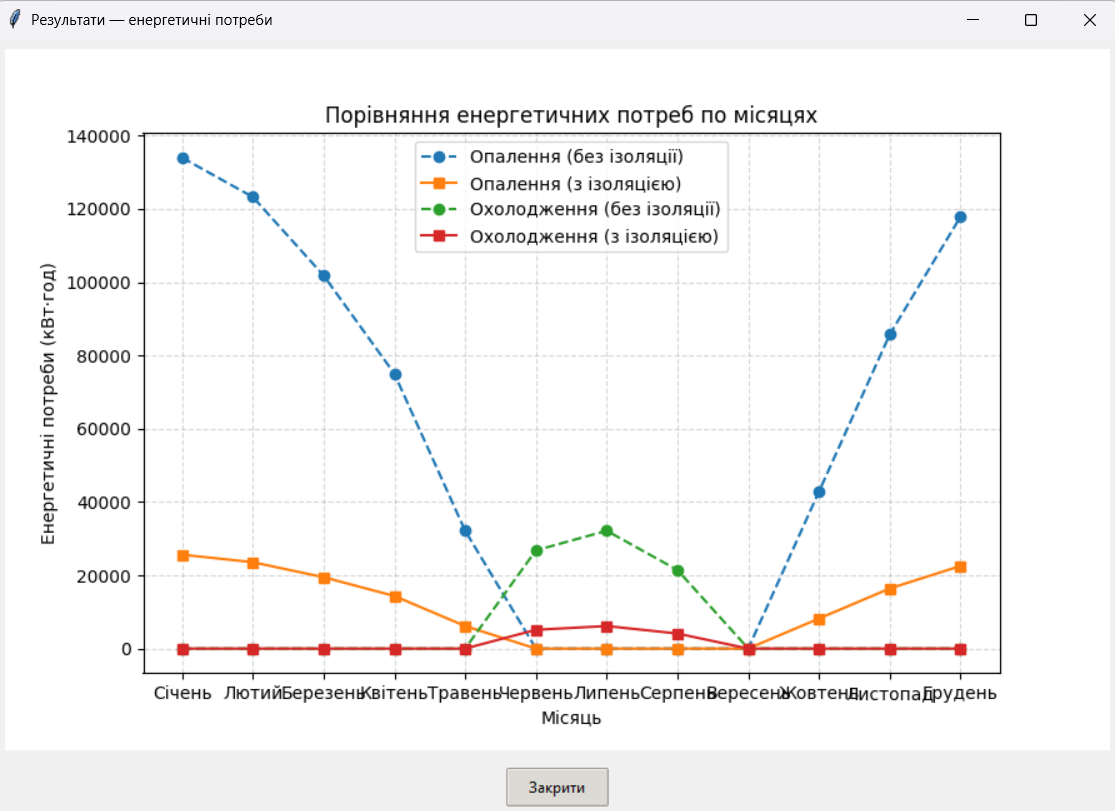
Для охолодження враховували тільки надмірну теплову енергію

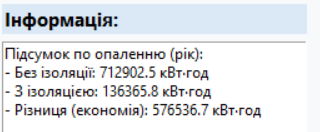
**Результати розрахунків**

1. **Дефолтні параметри:**

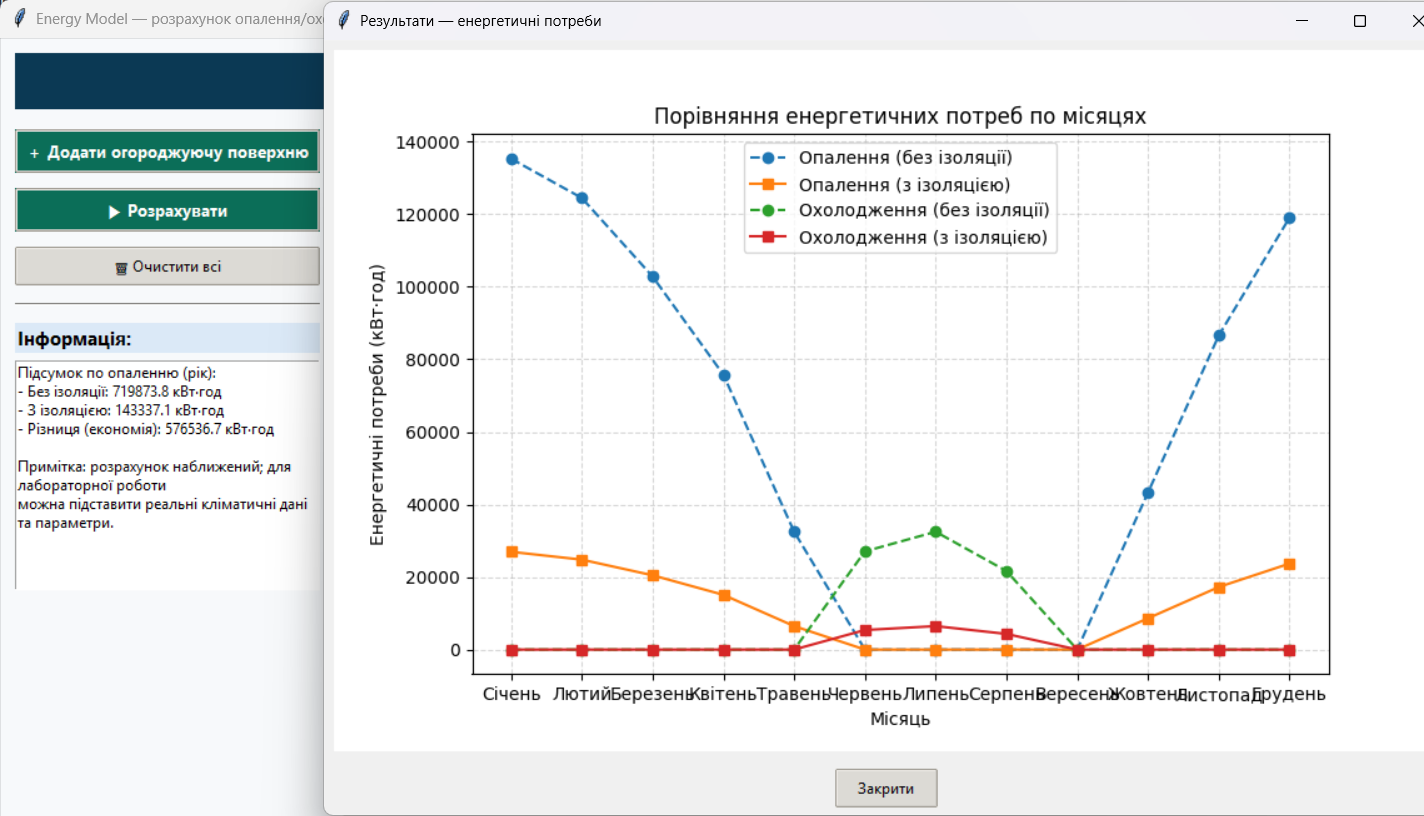
****

****

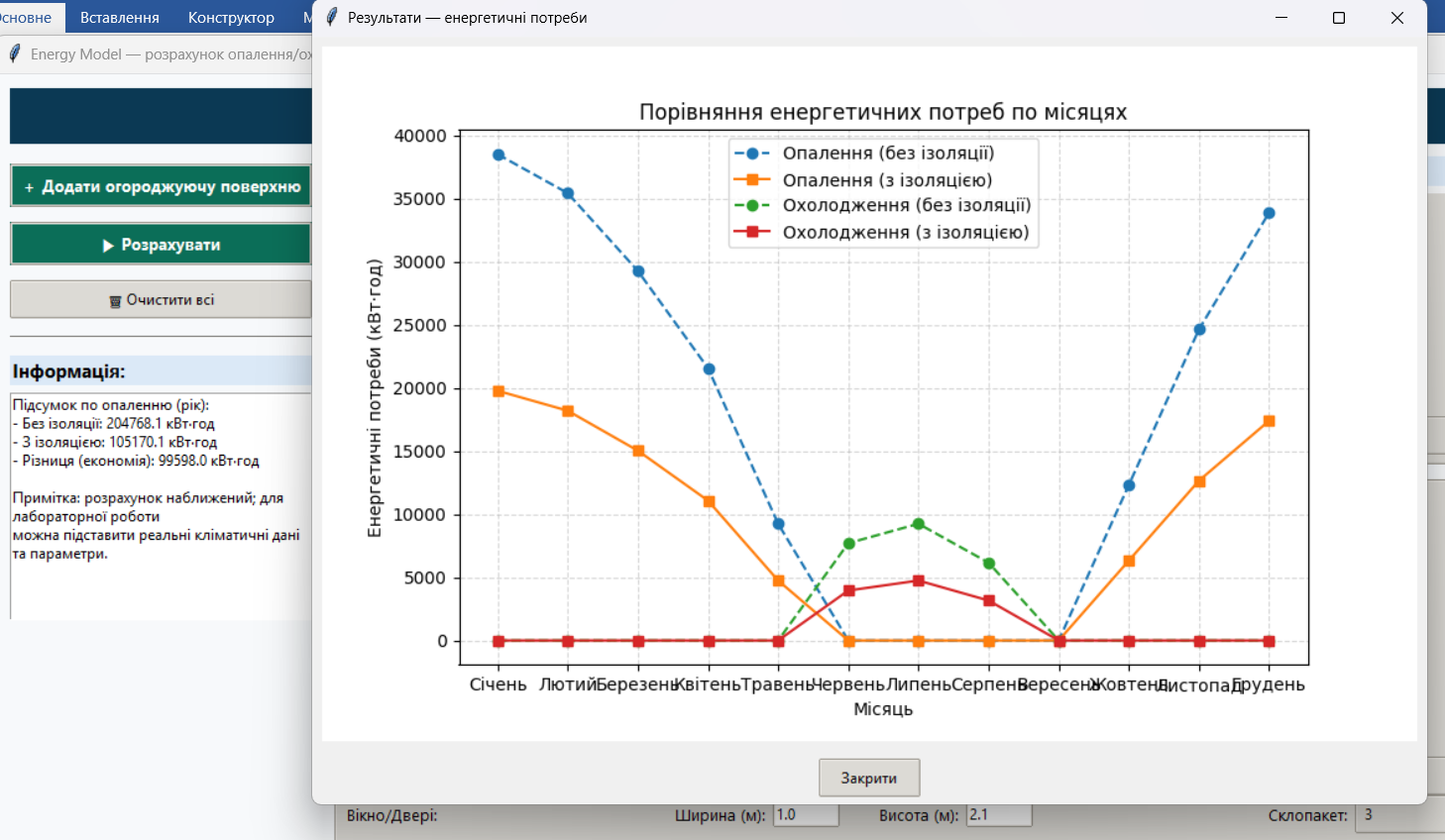
****

****

1. **Заміна вікон: спробуємо замінити склопакети у вікнах з 2 на 3, а також змінити газ з air на argon**

****

1. **Утеплення поверхонь: додамо +0.05 см стекловати (мінеральна вата) на кожну стіну**

****

**Висновок:**

У результаті виконання лабораторної роботи було створено програму розрахунку енергопотреб житлових приміщень на опалення/охолодження. В рамках вхідних даних було обрано реальний будинок з приблизними параметрами та можливими наслідками по витратам на опалення будинку. Результат показав, що встановлення теплоізоляції (на прикладі скловати) та/або використанні більш шільного стеклопакету дозволяє суттєво знизити витрати на енергоресурси для опалення приміщення, а також дозволяє покращити процес охолодження в теплий період часу.

**Лістинг програми:**

import numpy as np  
import tkinter as tk  
import matplotlib.pyplot as plt  
  
from tkinter import ttk, messagebox  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
  
MATERIAL\_CATALOG = [  
 {"name": "Цегла кераміка", "conductivity": 0.81},  
 {"name": "Цегла силікат", "conductivity": 0.87},  
 {"name": "Бетон", "conductivity": 1.86},  
 {"name": "Пінопласт", "conductivity": 0.04},  
 {"name": "Мінеральна вата", "conductivity": 0.045},  
 {"name": "Штукатурка цементна", "conductivity": 0.93},  
 {"name": "Штукатурка внутр. облицювання", "conductivity": 0.85},  
]  
  
GLASS\_UNIT\_UVALUES = {  
 "air": 0.35,  
 "argon": 0.25,  
 "krypton": 0.20  
}  
  
MONTH\_LABELS = ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень', 'Липень', 'Серпень', 'Вересень',  
 'Жовтень', 'Листопад', 'Грудень']  
  
MONTHLY\_TEMPERATURES = [-5, -3, 1, 6, 14, 30, 31, 29, 23, 12, 4, -2] # °C  
  
  
class EnergyModelApp:  
 def \_\_init\_\_(self, master: tk.Tk):  
 self.master = master  
 self.master.title("Energy Model — розрахунок опалення/охолодження")  
 self.master.geometry("980x640")  
 self.master.minsize(860, 560)  
  
 self.enclosures = []  
 self.\_configure\_style()  
 self.\_build\_interface()  
  
 def \_configure\_style(self):  
 style = ttk.Style()  
 style.theme\_use('clam')  
  
 # стиль кнопок/рамок  
 style.configure('App.TFrame', background='#f7f9fb')  
 style.configure('Header.TLabel', font=('Segoe UI', 14, 'bold'), background='#0b3954', foreground='white',  
 padding=8)  
 style.configure('Section.TLabel', font=('Segoe UI', 11, 'bold'), background='#dbe9f7')  
 style.configure('Small.TLabel', font=('Segoe UI', 10))  
 style.configure('Accent.TButton', font=('Segoe UI', 10, 'bold'), background='#0b6e58', foreground='white',  
 padding=6)  
 style.map('Accent.TButton', background=[('active', '#0a5d47')])  
  
 def \_build\_interface(self):  
 container = ttk.Frame(self.master, style='App.TFrame', padding=12)  
 container.pack(fill='both', expand=True)  
  
 # Header  
 header = ttk.Label(container, text="Калькулятор енергетичних потреб будівлі", style='Header.TLabel',  
 anchor='center')  
 header.pack(fill='x', pady=(0, 10))  
  
 # Controls frame (ліворуч)  
 controls\_frame = ttk.Frame(container, style='App.TFrame')  
 controls\_frame.pack(side='left', fill='y', padx=(0, 12))  
  
 add\_btn = ttk.Button(controls\_frame, text="＋ Додати огороджуючу поверхню", style='Accent.TButton',  
 command=self.add\_enclosure)  
 add\_btn.pack(fill='x', pady=6)  
  
 calc\_btn = ttk.Button(controls\_frame, text="▶ Розрахувати", style='Accent.TButton', command=self.calculate)  
 calc\_btn.pack(fill='x', pady=6)  
  
 clear\_btn = ttk.Button(controls\_frame, text="🗑 Очистити всі", command=self.clear\_all)  
 clear\_btn.pack(fill='x', pady=6)  
  
 ttk.Separator(controls\_frame, orient='horizontal').pack(fill='x', pady=8)  
  
 # Інфопанель  
 info\_label = ttk.Label(controls\_frame, text="Інформація:", style='Section.TLabel')  
 info\_label.pack(fill='x', pady=(6, 2))  
  
 self.summary\_text = tk.Text(controls\_frame, height=12, width=34, wrap='word', font=('Segoe UI', 9))  
 self.summary\_text.pack(fill='both', expand=False, pady=4)  
 self.summary\_text.insert('end',  
 "Додайте огороджуючі поверхні та матеріали.\nНатисніть 'Розрахувати' для перегляду графіків.")  
 self.summary\_text.configure(state='disabled')  
  
 # Area for walls/windows  
 editor\_frame = ttk.Frame(container, style='App.TFrame')  
 editor\_frame.pack(side='right', fill='both', expand=True)  
  
 editor\_header = ttk.Label(editor\_frame, text="Перелік огороджуючих поверхонь", style='Section.TLabel',  
 anchor='w')  
 editor\_header.pack(fill='x', pady=(0, 6))  
  
 # Canvas + Scrollbar для довгого списку стін  
 self.canvas = tk.Canvas(editor\_frame, background='#ffffff', highlightthickness=0)  
 self.scrollbar = ttk.Scrollbar(editor\_frame, orient='vertical', command=self.canvas.yview)  
 self.canvas.configure(yscrollcommand=self.scrollbar.set)  
  
 self.scrollbar.pack(side='right', fill='y')  
 self.canvas.pack(side='left', fill='both', expand=True)  
  
 self.inner\_frame = ttk.Frame(self.canvas, style='App.TFrame')  
 self.canvas.create\_window((0, 0), window=self.inner\_frame, anchor='nw')  
  
 self.inner\_frame.bind("<Configure>", lambda e: self.canvas.configure(scrollregion=self.canvas.bbox("all")))  
  
 # Операції з моделлю  
 def add\_enclosure(self):  
 enclosure = {  
 "width\_var": tk.DoubleVar(value=0.0),  
 "height\_var": tk.DoubleVar(value=0.0),  
 "layers": [],  
 "openings": []  
 }  
 index = len(self.enclosures)  
 self.enclosures.append(enclosure)  
  
 block = ttk.Frame(self.inner\_frame, relief='ridge', padding=8)  
 block.grid(row=index, column=0, sticky='ew', pady=6, padx=6)  
 block.columnconfigure(0, weight=1)  
  
 title = ttk.Label(block, text=f"Огороджуюча поверхня №{index + 1}", style='Section.TLabel')  
 title.grid(row=0, column=0, columnspan=6, sticky='w', pady=(0, 6))  
  
 ttk.Label(block, text="Ширина (м):", style='Small.TLabel').grid(row=1, column=0, sticky='e')  
 ttk.Entry(block, textvariable=enclosure["width\_var"], width=10).grid(row=1, column=1, sticky='w', padx=4)  
  
 ttk.Label(block, text="Висота (м):", style='Small.TLabel').grid(row=1, column=2, sticky='e')  
 ttk.Entry(block, textvariable=enclosure["height\_var"], width=10).grid(row=1, column=3, sticky='w', padx=4)  
  
 add\_layer\_btn = ttk.Button(block, text="＋ Додати шар матеріалу", command=lambda idx=index: self.add\_layer(idx))  
 add\_layer\_btn.grid(row=1, column=4, padx=6)  
  
 add\_opening\_btn = ttk.Button(block, text="＋ Додати вікно/двері",  
 command=lambda idx=index: self.add\_opening(idx))  
 add\_opening\_btn.grid(row=1, column=5, padx=6)  
  
 enclosure["ui\_block"] = block  
  
 def add\_layer(self, enclosure\_index: int):  
 enc = self.enclosures[enclosure\_index]  
 layer = {  
 "material\_var": tk.StringVar(value=MATERIAL\_CATALOG[0]["name"]),  
 "thickness\_var": tk.DoubleVar(value=0.1)  
 }  
 enc["layers"].append(layer)  
  
 parent = enc["ui\_block"]  
 row = 2 + len(enc["layers"]) + len(enc["openings"]) - 1 # простий індекс розміщення  
  
 ttk.Label(parent, text="Матеріал:", style='Small.TLabel').grid(row=row, column=0, sticky='e', pady=2)  
 material\_menu = ttk.OptionMenu(parent, layer["material\_var"], MATERIAL\_CATALOG[0]["name"],  
 \*[m["name"] for m in MATERIAL\_CATALOG])  
 material\_menu.grid(row=row, column=1, sticky='w', padx=4)  
  
 ttk.Label(parent, text="Товщина (м):", style='Small.TLabel').grid(row=row, column=2, sticky='e')  
 ttk.Entry(parent, textvariable=layer["thickness\_var"], width=10).grid(row=row, column=3, sticky='w', padx=4)  
  
 def add\_opening(self, enclosure\_index: int):  
 enc = self.enclosures[enclosure\_index]  
 opening = {  
 "width\_var": tk.DoubleVar(value=1.5),  
 "height\_var": tk.DoubleVar(value=1.4),  
 "packs\_var": tk.IntVar(value=2),  
 "gas\_var": tk.StringVar(value="air")  
 }  
 enc["openings"].append(opening)  
  
 parent = enc["ui\_block"]  
 row = 2 + len(enc["layers"]) + len(enc["openings"]) - 1  
  
 ttk.Label(parent, text="Вікно/Двері:", style='Small.TLabel').grid(row=row, column=0, sticky='e', pady=2)  
 ttk.Label(parent, text="Ширина (м):", style='Small.TLabel').grid(row=row, column=1, sticky='e')  
 ttk.Entry(parent, textvariable=opening["width\_var"], width=8).grid(row=row, column=2, sticky='w', padx=4)  
 ttk.Label(parent, text="Висота (м):", style='Small.TLabel').grid(row=row, column=3, sticky='e')  
 ttk.Entry(parent, textvariable=opening["height\_var"], width=8).grid(row=row, column=4, sticky='w', padx=4)  
  
 ttk.Label(parent, text="Склопакет:", style='Small.TLabel').grid(row=row, column=5, sticky='e')  
 ttk.OptionMenu(parent, opening["packs\_var"], 2, 1, 2, 3).grid(row=row, column=6, sticky='w', padx=4)  
  
 ttk.Label(parent, text="Газ:", style='Small.TLabel').grid(row=row, column=7, sticky='e')  
 ttk.OptionMenu(parent, opening["gas\_var"], "air", "air", "argon", "krypton").grid(row=row, column=8, sticky='w',  
 padx=4)  
  
 def clear\_all(self):  
 if not self.enclosures:  
 return  
 if not messagebox.askyesno("Підтвердження", "Ви впевнені, що бажаєте видалити всі поверхні?"):  
 return  
 for enc in self.enclosures:  
 try:  
 enc["ui\_block"].destroy()  
 except Exception:  
 pass  
 self.enclosures = []  
 self.summary\_text.configure(state='normal')  
 self.summary\_text.delete('1.0', 'end')  
 self.summary\_text.insert('end', "Очищено. Додайте нові поверхні.")  
 self.summary\_text.configure(state='disabled')  
  
 # --- Розрахунки ----  
 def \_compute\_heat\_losses(self, include\_insulation: bool):  
  
 results = []  
  
 for enc in self.enclosures:  
 width = enc["width\_var"].get()  
 height = enc["height\_var"].get()  
 area = width \* height  
  
 try:  
 r\_total = sum([  
 layer["thickness\_var"].get() / next(  
 m["conductivity"] for m in MATERIAL\_CATALOG if m["name"] == layer["material\_var"].get())  
 for layer in enc["layers"]  
 ])  
 except StopIteration:  
 messagebox.showwarning("Матеріал не знайдено", "Окремі шари мають невідомий матеріал — ігноруються.")  
 r\_total = 0.0  
  
 if include\_insulation:  
 r\_total += 0.1 / 0.045 # 10 см мінеральної вати  
  
 # Втрати через вікна/двері (приблизна формула U\_value \* area)  
 opening\_loss = sum([  
 (opening["width\_var"].get() \* opening["height\_var"].get()) / (  
 GLASS\_UNIT\_UVALUES[opening["gas\_var"].get()] \* opening["packs\_var"].get())  
 for opening in enc["openings"]  
 ])  
  
 # Підсумкова питомість втрат аналогічно до попередньої реалізації:  
 total\_loss = area / r\_total + opening\_loss if r\_total != 0 else opening\_loss  
 results.append(total\_loss)  
  
 return results  
  
 def \_monthly\_energy(self, loss\_values):  
heating = [  
 sum(loss\_values) \* 24 \* 30 \* max(0, (20 - t)) \* 0.024  
 for t in MONTHLY\_TEMPERATURES  
 ]  
 cooling = [  
 sum(loss\_values) \* 24 \* 30 \* max(0, (t - 25)) \* 0.024  
 for t in MONTHLY\_TEMPERATURES  
 ]  
 return heating, cooling  
  
 def calculate(self):  
 if not self.enclosures:  
 messagebox.showinfo("Пусто", "Додайте хоча б одну огороджуючу поверхню для розрахунку.")  
 return  
  
 losses\_no\_ins = self.\_compute\_heat\_losses(include\_insulation=False)  
 losses\_with\_ins = self.\_compute\_heat\_losses(include\_insulation=True)  
  
 heat\_no, cool\_no = self.\_monthly\_energy(losses\_no\_ins)  
 heat\_with, cool\_with = self.\_monthly\_energy(losses\_with\_ins)  
  
 self.\_display\_results(heat\_no, heat\_with, cool\_no, cool\_with)  
  
 def \_display\_results(self, heat\_no, heat\_with, cool\_no, cool\_with):  
 win = tk.Toplevel(self.master)  
 win.title("Результати — енергетичні потреби")  
 win.geometry("900x620")  
  
 fig, ax = plt.subplots(figsize=(9, 5))  
 months = MONTH\_LABELS  
  
 # Використовую ті ж формули і подання, кольори стилізував трохи  
 ax.plot(months, heat\_no, label="Опалення (без ізоляції)", linestyle="--", marker='o')  
 ax.plot(months, heat\_with, label="Опалення (з ізоляцією)", linestyle="-", marker='s')  
 ax.plot(months, cool\_no, label="Охолодження (без ізоляції)", linestyle="--", marker='o')  
 ax.plot(months, cool\_with, label="Охолодження (з ізоляцією)", linestyle="-", marker='s')  
  
 ax.set\_title("Порівняння енергетичних потреб по місяцях")  
 ax.set\_xlabel("Місяць")  
 ax.set\_ylabel("Енергетичні потреби (кВт·год)")  
 ax.legend()  
 ax.grid(True, linestyle='--', alpha=0.5)  
  
 canvas = FigureCanvasTkAgg(fig, master=win)  
 canvas.draw()  
 canvas.get\_tk\_widget().pack(fill='both', expand=True, padx=8, pady=8)  
  
 close\_btn = ttk.Button(win, text="Закрити", command=win.destroy)  
 close\_btn.pack(pady=6)  
  
 self.\_update\_summary(heat\_no, heat\_with)  
  
 def \_update\_summary(self, heat\_no, heat\_with):  
 total\_no = sum(heat\_no)  
 total\_with = sum(heat\_with)  
 saved = total\_no - total\_with  
  
 txt = (  
 f"Підсумок по опаленню (рік):\n"  
 f"- Без ізоляції: {total\_no:.1f} кВт·год\n"  
 f"- З ізоляцією: {total\_with:.1f} кВт·год\n"  
 f"- Різниця (економія): {saved:.1f} кВт·год\n\n"   
 )  
 self.summary\_text.configure(state='normal')  
 self.summary\_text.delete('1.0', 'end')  
 self.summary\_text.insert('end', txt)  
 self.summary\_text.configure(state='disabled')  
  
  
def main():  
 root = tk.Tk()  
 app = EnergyModelApp(root)  
 root.mainloop()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()