Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1
з дисципліни
«Об'єктно-Орієнтоване Програмування»

Виконала:
студентка групи ІМ-43:
Козаченко Софія Олександрівна

Перевірив:
Порєв В. М.

Київ 2025

Варіант завдання :

Номер 0 : Вікно діалогу для вводу тексту, яке має стрічку вводу (Edit Control) та дві кнопки: [Так] і [Відміна]. Якщо ввести рядок тексту і натиснути [Так], то у головному вікні повинен відображатися текст, що був введений.

Номер 1 : Вікно діалогу з повзуном горизонтального скролінгу (Horizontal scroll Bar) та дві кнопки: [Так] і [Відміна]. Рухаючи повзунок скролінгу користувач вводить число у діапазоні від 1 до 100. Після натискування кнопки [Так] вибране число буде відображатися у головному вікні.

Вихідний текст головного файлу програми :

```
import { app, BrowserWindow, ipcMain } from 'electron';
import path from 'path';
import { fileURLToPath } from 'url';
import Func_MOD1 from './mod1.js';
import Func_MOD2 from './mod2.js';

const __dirname = path.dirname(fileURLToPath(import.meta.url));
let mainWin;

function createMain() {
 mainWin = new BrowserWindow({
   width: 720,
   height: 520,
   webPreferences: {
     nodeIntegration: true,
     contextIsolation: false
   }
 });

 mainWin.loadFile(path.join(__dirname, 'index.html'));

 ipcMain.on('choose-work', async (event, payload) => {
   const { work } = payload || {};
   if (work === 'work1') {
     const code = await Func_MOD1(mainWin);
     console.log('Func_MOD1 returned', code);
   } else if (work === 'work2') {
     const code = await Func_MOD2(mainWin);
     console.log('Func_MOD2 returned', code);
   }
   if (!mainWin.isDestroyed()) mainWin.webContents.send('show-screen', 'menu');
 });
```

```
}

app.whenReady().then(createMain);
app.on('window-all-closed', () => app.quit());
```

Вихідні тексти модулів програми :

mod1.js

```
import { ipcMain } from 'electron';

export default function Func_MOD1(win) {
 const wc = win.webContents;
 wc.send('show-screen', 'dialog1');
 wc.send('mod1:focus-input');

 return new Promise((resolve) => {
   function onDialogAction(event, payload) {
     if (event.sender !== wc) return;
     const { action, text } = payload || {};
     if (action === 'yes') {
       wc.send('mod1:display-text', String(text || ''));
       cleanup();
       resolve(1);
     } else if (action === 'cancel') {
       cleanup();
       resolve(0);
     }
   }

   function cleanup() {
     try {
       ipcMain.removeListener('mod1:dialog-action', onDialogAction);
     } catch {}
   }

   ipcMain.on('mod1:dialog-action', onDialogAction);
   win.once('closed', () => {
     cleanup();
     resolve(0);
   });
 });
}
```

mod2.js

```
import { ipcMain } from 'electron';

export default function Func_MOD2(win) {
 const wc = win.webContents;
 wc.send('show-screen', 'dialog2');
 wc.send('mod2:focus-input');

 return new Promise((resolve) => {
   function onAction(event, payload) {
     if (event.sender !== wc) return;
     const { action, number } = payload || {};
     if (action === 'yes') {
       wc.send('mod2:display-number', number);
       cleanup();
       resolve(1);
     } else if (action === 'cancel') {
       cleanup();
       resolve(0);
     }
   }

   function cleanup() {
     try { ipcMain.removeListener('mod2:action', onAction); } catch {}
   }

   ipcMain.on('mod2:action', onAction);
   win.once('closed', () => { cleanup(); resolve(0); });
 });
}
```

index.html

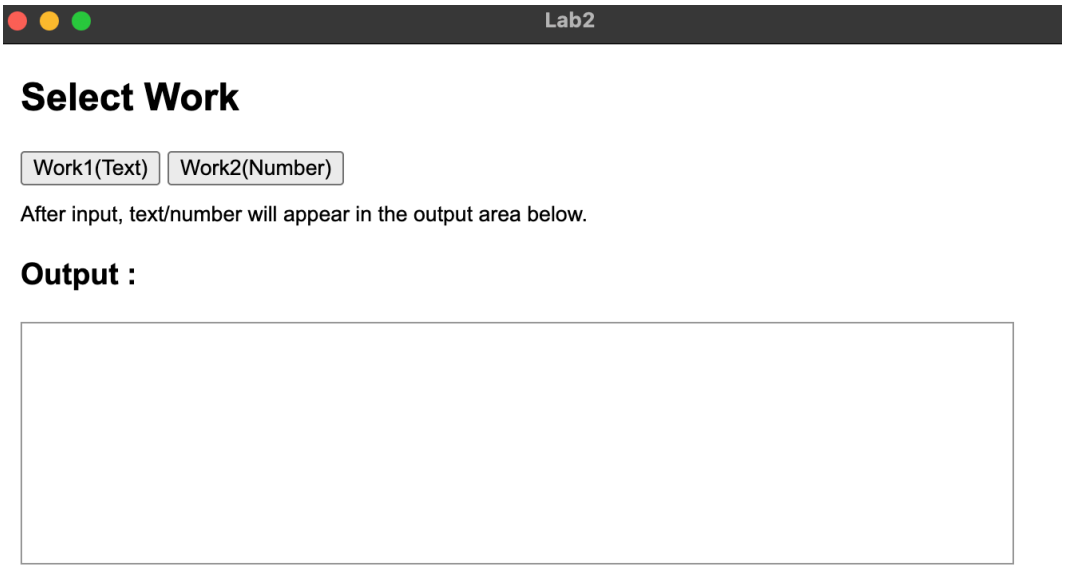
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8" />
 <title>Lab2</title>
 <style>
   body { font-family: Arial, sans-serif; margin: 16px; }
   .screen { display: none; }
   .active { display: block; }
```

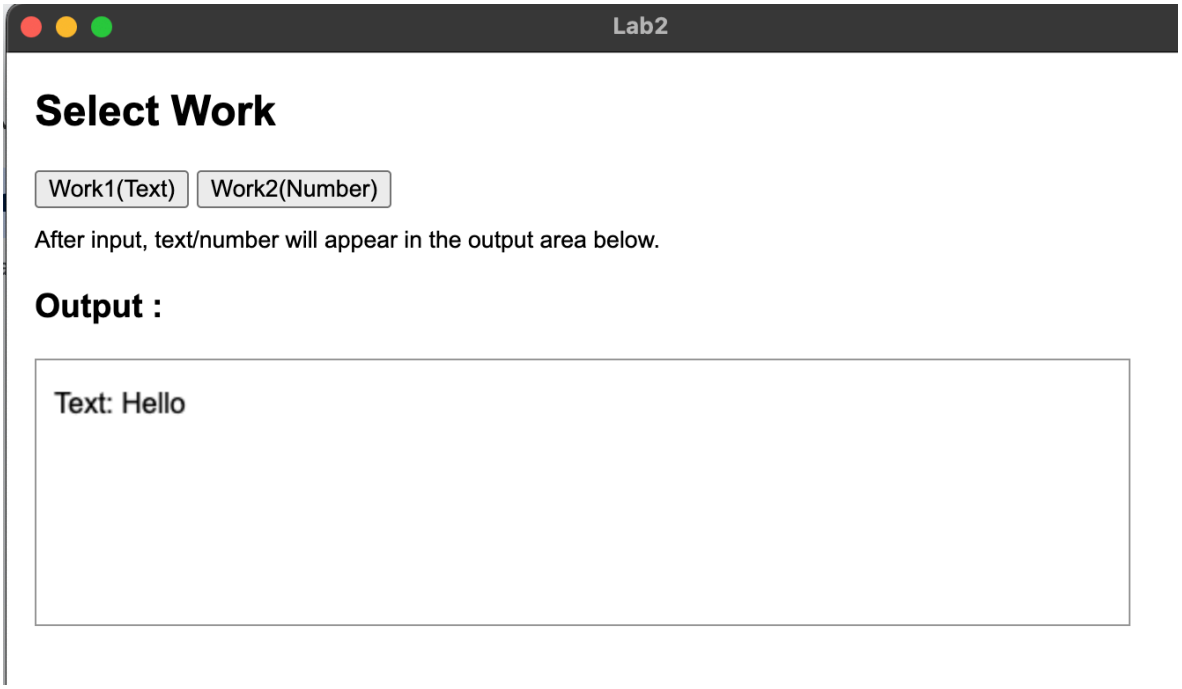
```
  .row { margin: 8px 0; }
  canvas { border: 1px solid #999; display: block; margin-top: 10px; }
  label { display: inline-block; width: 120px; }
 </style>
</head>
<body>
 <div id="menu" class="screen active">
  <h2>Select Work</h2>
  <div class="row">
   <button id="work1">Work1(Text)</button>
   <button id="work2">Work2(Number)</button>
  </div>
  <div class="row">
   <small>After input, text/number will appear in the output area below.</small>
  </div>
  <h3>Output :</h3>
  <canvas id="outputCanvas" width="620" height="150"></canvas>
 </div>

 <div id="dialog1" class="screen">
  <h2>Text Input Dialog</h2>
  <div class="row">
   <label for="editInput">Enter text:</label>
   <input id="editInput" type="text" style="width:400px;" />
  </div>
  <div class="row">
   <button id="btnYes">OK</button>
   <button id="btnCancel">Cancel</button>
  </div>
 </div>

 <div id="dialog2" class="screen">
  <h2>Number Input Dialog</h2>
  <div class="row">
   <label for="numRange">Choose number:</label>
   <input id="numRange" type="range" min="1" max="100" value="50" />
   <span id="numValue">50</span>
  </div>
  <div class="row">
   <button id="btnNumYes">OK</button>
   <button id="btnNumCancel">Cancel</button>
  </div>
 </div>

 <script>
```

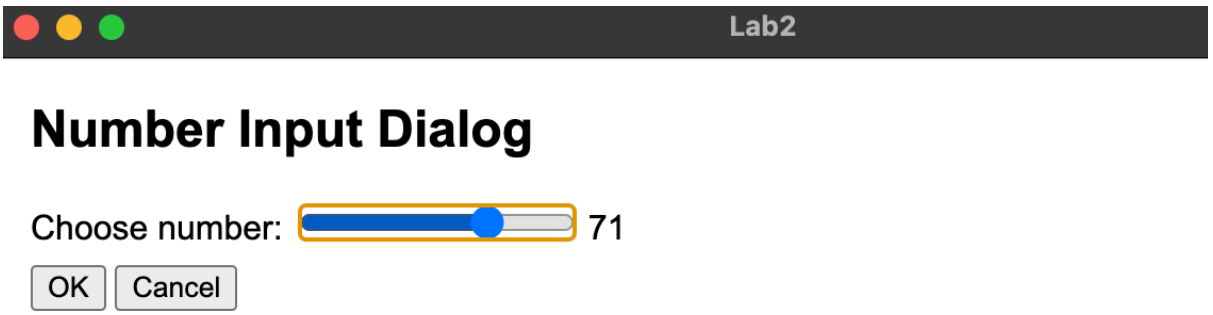
```
const { ipcRenderer } = require('electron');

const screens = {
  menu: document.getElementById('menu'),
  dialog1: document.getElementById('dialog1'),
  dialog2: document.getElementById('dialog2')
};

let storedText = "";
let storedNumber = null;

function show(screen) {
  Object.values(screens).forEach(s => s.classList.remove('active'));
  if (screens[screen]) screens[screen].classList.add('active');
}

function redrawCanvas() {
  const canvas = document.getElementById('outputCanvas');
  const ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  ctx.font = '16px Arial';

  let y = 30;
  if (storedText) {
    ctx.fillText("Text: " + storedText, 10, y);
    y += 30;
  }
  if (storedNumber !== null) {
    ctx.fillText("Number: " + storedNumber, 10, y);
  }
}

document.getElementById('work1').onclick = () => {
  ipcRenderer.send('choose-work', { work: 'work1' });
};
document.getElementById('work2').onclick = () => {
  ipcRenderer.send('choose-work', { work: 'work2' });
};

document.getElementById('btnYes').onclick = () => {
  const txt = document.getElementById('editInput').value || '';
  ipcRenderer.send('mod1:dialog-action', { action: 'yes', text: txt });
};
document.getElementById('btnCancel').onclick = () => {
  ipcRenderer.send('mod1:dialog-action', { action: 'cancel' });
```

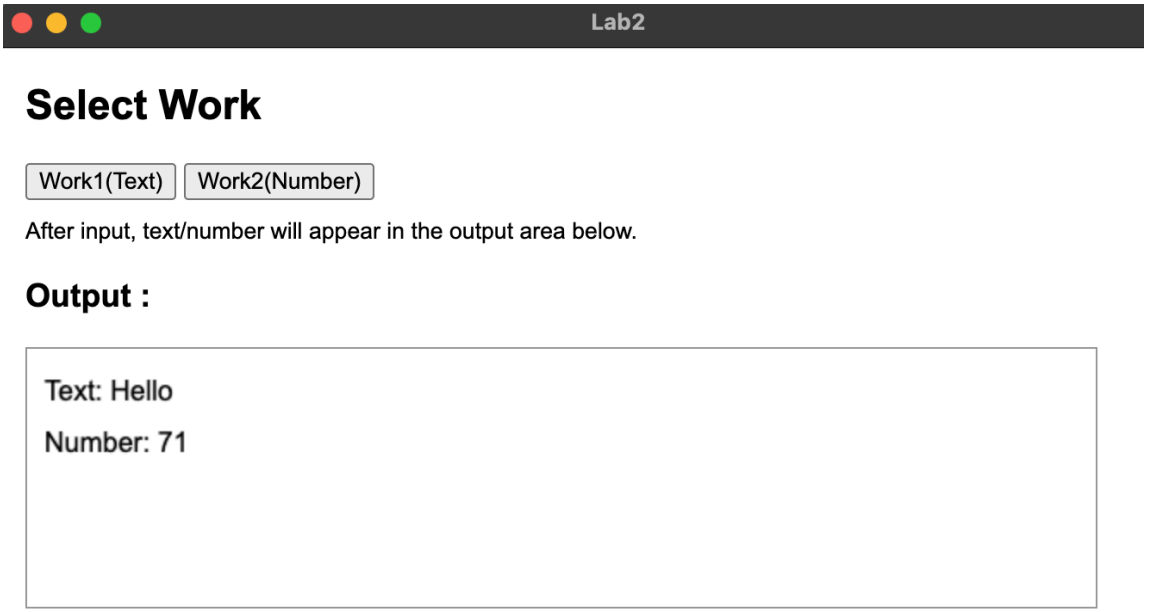
```
  };

  const numRange = document.getElementById('numRange');
  const numValue = document.getElementById('numValue');
  numRange.oninput = () => {
    numValue.innerText = numRange.value;
  };

  document.getElementById('btnNumYes').onclick = () => {
    ipcRenderer.send('mod2:action', { action: 'yes', number:
parseInt(numRange.value) });
  };
  document.getElementById('btnNumCancel').onclick = () => {
    ipcRenderer.send('mod2:action', { action: 'cancel' });
  };

  ipcRenderer.on('show-screen', (ev, screen) => show(screen));

  ipcRenderer.on('mod1:display-text', (ev, text) => {
    storedText = text || "";
    redrawCanvas();
  });

  ipcRenderer.on('mod1:focus-input', () => {
    show('dialog1');
    setTimeout(() => {
      document.getElementById('editInput').focus();
    }, 5);
  });

  ipcRenderer.on('mod2:display-number', (ev, number) => {
    storedNumber = number;
    redrawCanvas();
  });

  ipcRenderer.on('mod2:focus-input', () => {
    show('dialog2');
  });

  show('menu');
 </script>
</body>
</html>
```

Скріншоти тестування програми :









Висновок :
Виконавши лабораторну роботу 1 я навчилась поєднувати теоретичні знання з практичним досвідом роботи у сучасному інструментарії розробника.

У даній лабораторній роботі було реалізовано завдання відповідно до варіантів 0 та 1. Замість традиційного підходу з використанням WinAPI на C++, для виконання роботи я обрала JavaScript з Node.js та Electron. Такий вибір зумовлений тим, що основна розробка велась на macOS, де безпосереднє використання WinAPI є неможливим. Electron дозволив відтворити необхідну логіку роботи вікон, діалогів та обробки подій.

У результаті виконання лабораторної я навчилася : застосовувати Electron для створення діалогових вікон та обробки подій, відтворювати

функціонал завдань WinAPI у кросплатформовому середовищі, організовувати програму у вигляді модулів з чітко визначеним інтерфейсом.