Київський національний університет імені Тараса Шевченка

Кононов М.В.
Веремій Ю.П.

Прикладне програмування в телекомунікаційних системах

Методичні вказівки до проведення лабораторних робіт

для студентів спеціальності
172 «Телекомунікації та радіотехніка»
факультету радіофізики, електроніки
та комп'ютерних систем

Київ – 2020

УДК 004

Прикладне програмування в телекомунікаційних системах. Методичні вказівки до проведення лабораторних робіт для студентів спеціальності 172 «Телекомунікації та радіотехніка» факультету радіофізики електроніки та комп'ютерних систем / Кононов М.В. Веремій Ю.П. – Київ: ФРЕКС Київського національного університету імені Тараса Шевченка, 2020. – 50 с.

Рецензенти:

завідувач кафедри фізичної електроніки Київського національного університету імені Тараса Шевченка, професор, доктор фізико-математичних наук Веклич Анатолій Миколайович,

доцент кафедри математики та теоретичної радіофізики Київського національного університету імені Тараса Шевченка, кандидат фізико-математичних наук Єфіменко Світлана Володимирівна.

Ухвалено Вченою радою
факультету радіофізики електроніки та комп'ютерних систем
Київського національного університету імені Тараса Шевченка
(протокол № 3 від 13.10.2020)

© Кононов М.В., Веремій Ю.П. 2020

Зміст

Вступ 4

Прикладне програмування 5

Особливості виконання практикуму 7

Лабораторна робота №1. Основи форматування тексту web-сторінки 10

Лабораторна робота №2. Формування запитів від клієнта до сервера 17

Лабораторна робота №3. Реалізація графічного інтерфейсу користувача на основі використання HTML та JavaScript 21

Лабораторна робота №4. Інтерактивна зміна інтерфейсу користувача на основі використання HTML разом з JavaScript 26

Лабораторна робота №5. Інтерактивна зміна інтерфейсу користувача на основі запиту до сервера .. 30

Лабораторна робота №6. Ускладнений варіант інтерактивної зміни інтерфейсу користувача на основі запиту до сервера 33

Лабораторна робота №7. Основні операції з базою даних 37

Лабораторна робота №8. Реалізація web-застосунку трирівневої архітектури 42

Лабораторна робота №9. Реалізація графічного інтерфейсу користувача «десктопного» застосунку 45

Література 50

Вступ

На перший погляд може здаватись, що технології програмування для спеціальності 172 (Телекомунікації та радіотехніка) є сторонніми. Однак треба враховувати, що практично усі сучасні технології телекомунікації є цифровими і спираються на програмну обробку даних. Радіотехніка, звичайно, використовує аналогові пристрої для генерування та підсилення електричних сигналів і узгодження цих елементів з середовищем передачі даних. Але сучасна радіоелектронна база також значною мірою спирається на цифровий режим та програмну обробку. Навіть серед антенних пристроїв найцікавішим є цифрова антенна решітка. Відповідно, як мінімум, фахівець з даної спеціальності повинен розуміти певні особливості реалізації програмної обробки інформації.

На першому курсі ви вивчали основи програмування, що надає певний фундамент для використання цифрової обробки даних, наприклад, для подальшого засвоєння програмування мікроконтролерів. Зазначимо, що специфічність роботи з мікроконтролерами спирається на відносно глибоке розуміння взаємодії програмованого пристрою з апаратним оточенням для роботи з окремими електричними сигналами. При цьому структура програми є відносно простою в порівнянні з іншими програмними задачами. Але, як при використанні у програмній обробці інформації мережевого сполучення, так і для залучення програмних засобів як частини телекомунікаційної системи, виникають значно складніші задачі: системні та прикладні.

В межах даної навчальної дисципліни розглядається саме прикладне програмування. Оскільки будь-яке програмування є практичною діяльністю, до лекційної частини доданий практикум, який передбачає засвоєння певних навичок розробки проектів прикладних програмних засобів. При цьому увага акцентується на мережевому режимі роботи прикладних застосунків.

Прикладне програмування

Прикладне програмування є процесом розробки програмних засобів, які виконують конкретні задачі користувачів. Відповідно одною з базових відмінностей таких засобів, наприклад від системних, є необхідність реалізації зручної взаємодії з користувачем, в тому числі непрофесіоналом в інформаційних технологіях. Останнє призвело до широкого розповсюдження графічної форми такого інтерфейсу – *Graphic User Interface (GUI).* Він спирається на використання елементів на екрані, які графічно відтворюють звичні для пересічної людини "фізичні" (апаратні) елементи (кнопки, панелі, індикатори тощо).

Історично склалось, що прикладні програми досить довго орієнтувались на взаємодію у певний час тільки з одним користувачам, оскільки локальний комп'ютер має одну клавіатуру та один монітор. Такий режим роботи дозволяє переглядати або редагувати документи у певному форматі, виконувати моделювання або іншу обробку даних, які містяться (або можуть бути зібрані) на комп'ютері користувача.

З часом через потребу забезпечення взаємодії користувачів між собою та доступу до віддалених даних крім використання локального режиму довелось залучити мережевий. Для реалізації такого режиму роботи застосунку зручно використовувати клієнт-серверну архітектуру. При цьому тільки частина задачі виконується на комп'ютері користувача (клієнті), інше виконується на спільному ресурсі (сервері).

Таким чином мережевий застосунок є складеним засобом, який передбачає взаємодію окремих його компонентів. При розробці прикладного проекту, звичайно, можна розробити "усе з нуля", але така розробка буде надто дорогою і довгою. Для зменшення її ресурсоємності використовують якомога більше готових складових (протоколів, форматів, бібліотек тощо). Для забезпечення мережевої взаємодії доцільно використати готовий сервер та засіб перегляду даних. Ця ідея

призвела до розширення використання web-технологій зі значно простішого випадку "павутинного" зв'язування статичних текстових документів до випадку складної обробки і відтворення динамічного стану певної системи.

Для забезпечення динамічної роботи з інформацією традиційні у web-технологіях web-сервер та web-браузер доповнюються відповідними підсистемами програмної обробки. У більш складних випадках (яких з часом все більше і більше) для забезпечення ефективного доступу до даних використовується окремий сервер бази даних.

Таким чином, найбільш поширеною (але не єдиною можливою) архітектурою застосунку для роботи у телекомунікаційніїй системі є web. Тому основою даного практикуму є саме цей варіант реалізації програмного засобу.

В межах даного практикуму розглянемо основи (нажаль, тільки основи) створення усіх частин такого багаторівневого застосунку і базові технології, які використовуються для цього. Одразу зауважимо, що при цьому виникають певні особливості використання програмування і відладки, хоча загальні ідеї працюють і тут, що дозволяє спиратись на раніше отримані знання та навички з розробки простих програмних проектів.

Web-технології вивчаються у восьми завданнях. Ще одна лабораторна робота присвячена загальним принципам створення графічного інтерфейсу користувача (GUI) для "десктопного" застосунку на основі використання графічного програмування за допомогою сучасного IDE. Зрозуміло, що в цьому випадку мережева складова залишається осторонь, але слід пам'ятати, що світ живе "не одним вебом" і подивитись таку архітектуру програмного продукту є також необхідним. Крім того "десктопний" застосунок дозволяє краще ознайомитись з реалізацією подіє-орієнтованої програми, яка значну частину часу роботи знаходиться в режимі очікування повідомлень про події (наприклад натискання мишею на зображення кнопки, зміна тексту у деякому полі тощо), на появу кожного з яких викликається своя функція-обробник.

Особливості виконання практикуму

Перш ніж перейти безпосередньо до завдань практикуму, трохи зупинимось на деяких особливостях виконання програмних проектів для створення мережевих прикладних програм.

Загалом довільну програму можна розробляти з використанням довільного текстового редактору та окремого компілятора. Але такий підхід не є зручним та ефективним. Саме тому вже на першому курсі використовувалось одне з інтегрованих середовищ розробки (IDE – Integrated Development Environment). Кожне IDE є пристосованим до певних технологій і відповідних мов програмування. Існують IDE і для web-розробки. Більш того, для таких проектів часто використовують специфічні фреймворки, які значною мірою нав'язують архітектурні особливості проекту. Є багато різних фреймворків, але усі вони спираються на використання певних мов програмування. Тобто навіть для роботи у фреймворках треба розуміти базовий рівень використання програмних технологій для web. Таким чином, за основу практикуму було обране використання "чистих" мов, які виконують обробку даних на боці клієнту та на сервері.

Для реалізації зовнішнього вигляду GUI, роль якого відіграє web-сторінка, використовується HTML разом із CSS. Сукупність тексту у відповідних форматах обробляється та відтворюється браузером, тобто сервер в даному випадку є тільки джерелом даних і редагуванням відповідного проекту можна займатись локально, взагалі без web-сервера, використовуючи тільки текстовий редактор та довільний браузер (є небажаним використовувати Internet Explorer від Microsoft).

Для редагування тексту зручно серед багатьох редакторів обрати безкоштовну програму Notepad++ (рис.1), яка відповідно до обраного формату автоматично виділяє кольорами певні семантичні частини тексту та має деякий рівень підказки набору

тексту. Все це значно спрощує роботу. Відповідно, цей редактор використовується у комп'ютерному класі, у якому проводиться практикум.

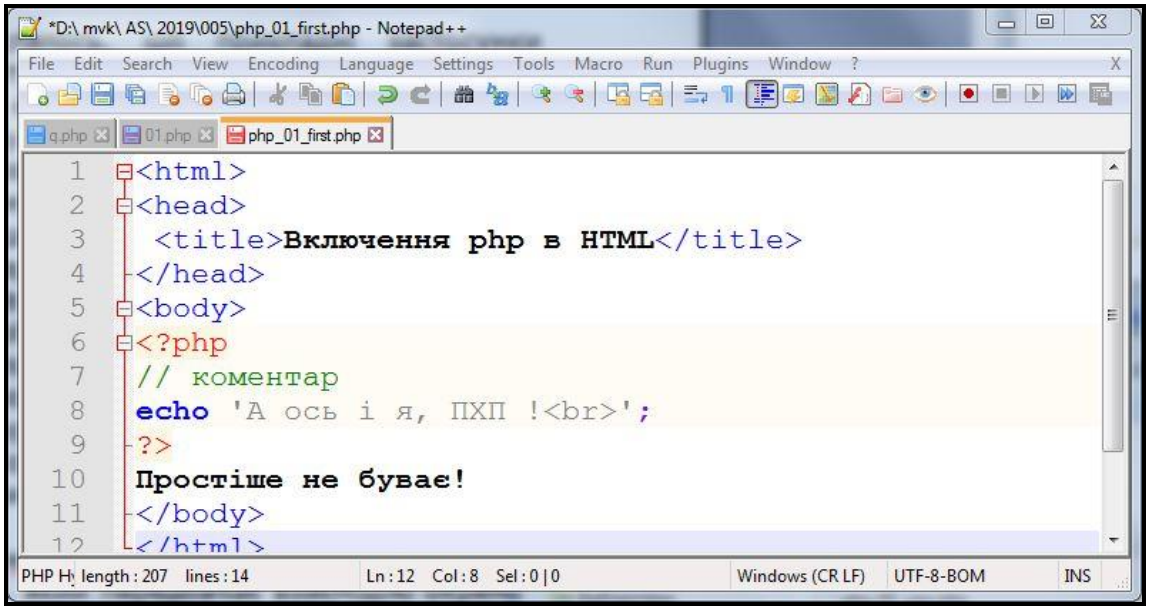


Рис.1

Оскільки в даній навчальній дисципліні мова йде не про статичні сторінки, а про деяку прикладну програму, зрозуміло, що додатково потрібна реалізація подієорієнтованості та можливості зміни вигляду сторінки. За це відповідає JavaScript, який виконується браузером. Розробка такого проекту також не вимагає обробки даних з серверного боку. Тобто і для такого проекту є достатнім використати текстовий редактор та браузер. При цьому вбудовані у браузер засоби навіть дозволяють досить зручно виконати відладку.

При написанні програми, що виконує обробку даних на сервері і звертається до серверу бази даних, відповідні тексти повинні бути у папках web-сервера і запускатись саме сервером. Для цього викладачем забезпечується мережевий доступ до відповідної папки на комп'ютері викладача, який відіграє роль сервера. Для кожного студента виділяється окрема мережева папка. До цього ж сервера є можливість звернення з браузера власного комп'ютера за його ім'ям у локальній мережі. Таким чином редагувати текст треба у своїй

мережевій папці, запускати проект можна тільки через браузер на власному комп'ютері.

Для зручності навчання і можливості практикуватись вдома рекомендується встановити на особистому комп'ютері один з варіантів комплексного пакету, який включає web-сервер, налаштований на підтримку певної мови програмування, та SQL-сервер. До таких засобів можна віднести, наприклад, WampServer, XAMPP, Денвер (усі вони є вільними). Звичайно, можна окремо встановити якийсь web-сервер, модуль підтримки мови програмування на сервері, один з SQL-серверів. Але це з самого початку вимагає певної кваліфікації, в тому числі навичок в адмініструванні.

У комп'ютерному класі використовується WampServer, встановлений на комп'ютері викладача, тобто комплекс з web-серверу Apache з модулем підтримки PHP та SQL-серверу MySQL. Дозволяється використовувати в класі власний комп'ютер з відповідною конфігурацією програмних засобів, наприклад, з одним із зазначених вище пакетів.

Зауважимо, що робота з довільним інженерним проектом (це стосується не тільки програмування) вимагає точного виконання наперед узгодженого технічного завдання. Для прикладних програмних проектів це передбачає в першу чергу забезпечення потрібної функціональності, але крім цього стосується і вигляду інтерфейсу користувача (або ж web-сторінки). Тобто при виконанні завдань практикуму слід звертати увагу навіть на такі "дрібниці".

Даний практикум не передбачає традиційного для більшості лабораторних практикумів оформлення письмового звіту. Виконання його робіт зараховується на основі максимально повного виконання завдання (перевіряється працездатність розробленого студентом проекту) та його захисту у співбесіді з викладачем.

Лабораторна робота №1. Основи форматування тексту web-сторінки

Мета роботи: навчитись задавати відтворення контенту web-документу завданням тегів, відповідних їх атрибутів та стилів.

Загальні відомості

Театр починається з вішалки (К. Станіславський). А прикладна програма із зручного та привабливого вигляду інтерфейсу користувача. Тому його дизайну завжди приділяють велику увагу, в тому числі, коли мова йде про web-застосунок. Саме тому першим завданням даного практикуму є відтворення у заданому вигляді деякого контенту.

Web-сторінка є не тільки способом відтворення деякої інформації (контенту), а й засобом зручної її подачі. Саме тому для її запису отримала розповсюдження спеціальна мова, яка до текстового запису контенту додає опис особливостей його відтворення. Загальна ідея такого накладання полягає у нав'язуванні певному фрагменту тексту деякої властивості. Для цього, зрозуміло, треба мати можливість вказування:

- характеристик властивості (стилю).
- границь застосування цієї властивості;

Щоб забезпечити такий запис була розроблена спеціальна мова розмітки тексту. З врахуванням потреби відтворення контенту з перехресними посиланнями було замість терміну "текст" використано "гіпертекст", що і сформувало назву HTML. Оскільки першочергово задача стосувалась саме роботи з текстом, спеціальні позначки виявилось слушним також зробити не бінарними, а виключно текстовими за рахунок накладання деяких обмежень не використання невеликої кількості спеціальних символів, основними з яких в цій мові були обрані кутові дужки (символи "більше" та "менше"). Вони

позначили службові вставки – теги, які в свою чергу визначають саму властивість і границі її накладення.

Певний час такого способу розмітки було цілком достатньо. Але необхідність з одного боку детально розписувати кожну з властивостей на великій кількості різних фрагментів сторінки, а з іншого боку потреба забезпечення єдиної стилістики оформлення багатосторінкового документу, призводить до необхідності винесення опису стилю у відокремлений фрагмент тексту. Це призвело до широкого використання іншої спеціальної мови – CSS (Cascading Style Sheets), використанням якої забезпечується таке відокремлення опису стилів і забезпечуються значно ширші можливості зміни відтворення контенту.

У CSS записується сукупність стилів, кожен з яких у свою чергу є списком правил. Кожне правило має один або декілька селекторів та список властивостей (через крапку з комою) у фігурних дужках. Визначення властивості описується її назвою та значенням (після двокрапки).

Крім накладання властивостей відтворення тексту та додаткових блоків даних, які задаються як текстове посилання, додатковою задачею є розміщення елементів на сторінці. Найбільш примітивним із таких способів є розміщення у рядок з перенесенням у наступний того, що вже не вміщується. До цього додається примусове перенесення рядка. Для відтворення тексту у багатьох випадках цього достатньо, але навіть для цього типу даних не завжди.

При реалізації графічного інтерфейсу користувача є потреба у більш складному розміщенні елементів. Для цього, наприклад, може бути використана таблиця. Хоча, одразу зазначимо, що це не єдиний спосіб, і зараз він вважається дещо застарілим через недостатню гнучкість. Але цей спосіб форматування є доволі простим і наочним. Крім того, не слід забувати про те, що взагалі таблиця є популярним способом відображення складної ділової інформації, в тому числі інженерної. Тому в завданні даної роботи за основу обрано саме таблицю.

Завдання

1. *Реалізувати таблицю за зразком зовнішнього вигляду.*
2. *У таблицю вставити вказані на зразку складні елементи.*
3. *Виконати переформатування сторінки зовнішню таблицею стилів.*

Хід виконання

1. Реалізація та форматування таблиці.

1.1. Запустити Notepad++, створити новий документ, записати його у робочу папку як сторінку у форматі html.

1.2. Написати текст «порожньої» заготовки web-сторінки (рис.2) .

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <title>Лаб робота 1</title>
5  <meta charset="UTF-8">
6  </head>
7  <body>
8
9  </body>
10 </html>
```

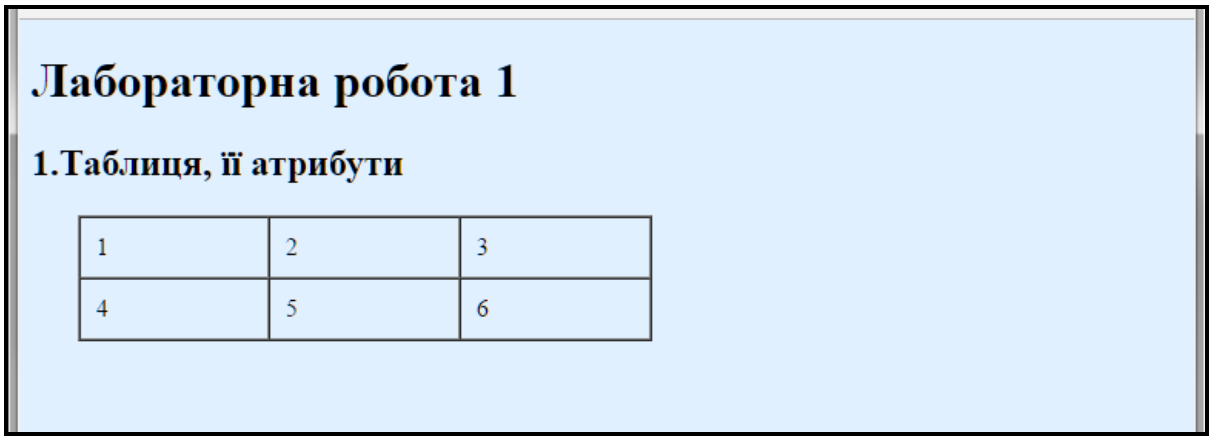
Рис.2

1.3. Завантажити сторінку у web-браузер.

Увага! Після зміни тексту у редакторі перед завантаженням або оновленням сторінки у браузері проект треба обов’язково записати.

1.4. Реалізувати просту таблицю за зразком рис.3. При цьому необхідно використати атрибути

тегу <table> (рис.4) – border, cellspacing, cellpadding (з'ясуйте, на що вони впливають).



Лабораторна робота 1

1.Таблиця, її атрибути

1	2	3
4	5	6

Рис.3

```
<table style="width:50%" border="1" >
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Рис.4 – Тег Table

1.5. З'ясувати, що зміниться, якщо таблицю охопити тегом <h3>.

1.6. Реалізувати ще одну таблицю за зразком рис.5.

Лабораторна робота 1

1.Таблиця, її атрибути

1	2	3
4	5	6

1	2	3
4	5	6

Рис.5

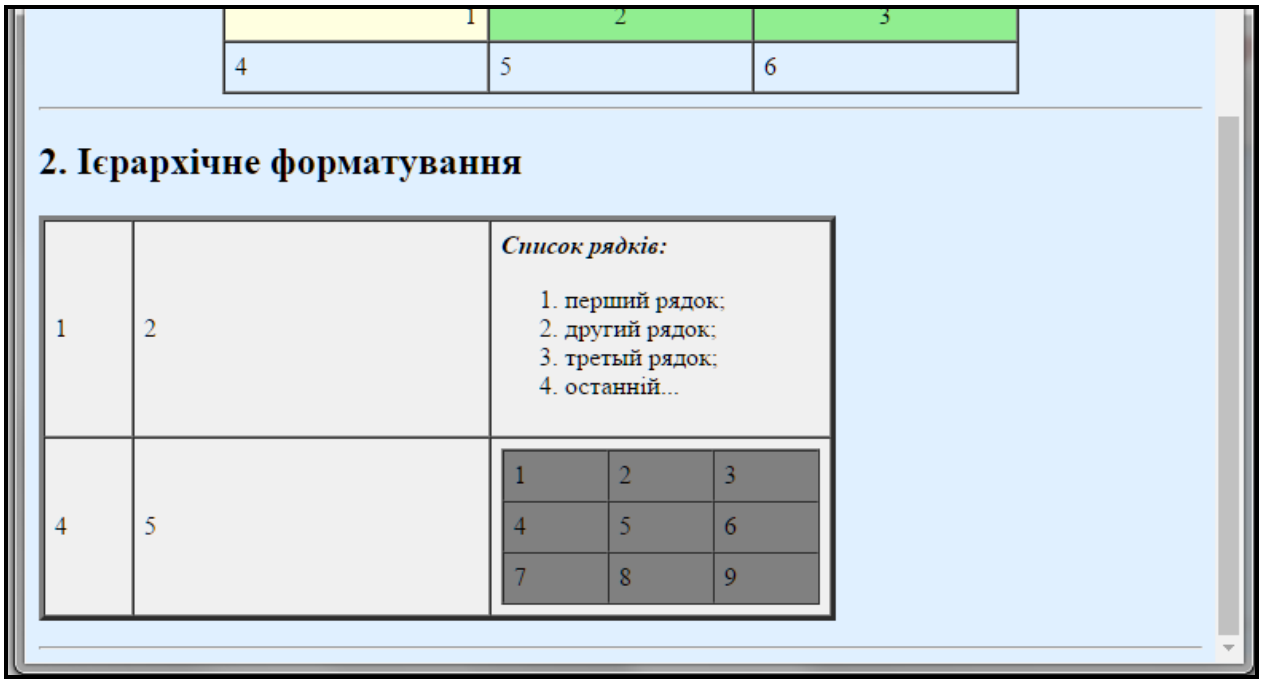
2. Вставка у таблицю складних елементів

2.1. Реалізувати просту таблицю, аналогічну за структурою п.1.4 (рис.3).

2.2. Вставити у відповідне поле таблиці список за зразком рис.6.

2.3. Вставити у відповідне поле таблиці таблицю за зразком рис.6.

2.4. Виконати форматування відповідно до рис.6.



1	2	3
4	5	6

2. Ієрархічне форматування

1	2	***Список рядків:*** 1. перший рядок; 2. другий рядок; 3. третий рядок; 4. останній...
4	5	1 2 3 4 5 6 7 8 9

Рис.6

3. Форматування таблицею стилів.

3.1. Створити окремий файл (css, рис.7) з описом потрібних форматів.

```
19    h3 {
20    /*font-family: "arial";*/
21    font-style:italic;
22    color:#0050a0;
23    font-size: 14pt;  /*1in = 2.54cm = 25.4mm = 72pt = 6pc*/
24    line-height: 2px; /*пікселі*/
25    margin-left: 0em;
26    margin-top: 1em; /* одиниці відступу, пропорційні розміру шрифту */
27    }
```

Рис.7 – Приклад фрагменту файлу стилів (CSS)

3.2. Приєднати файл форматування до раніше створеної сторінки. Приклад приєднання файлу з таблицею стилів:

```
<link rel="stylesheet" href="my_style.css" />
```

3.3. Редагуванням стилів у CSS файлі досягти форматування за зразком рис.8.

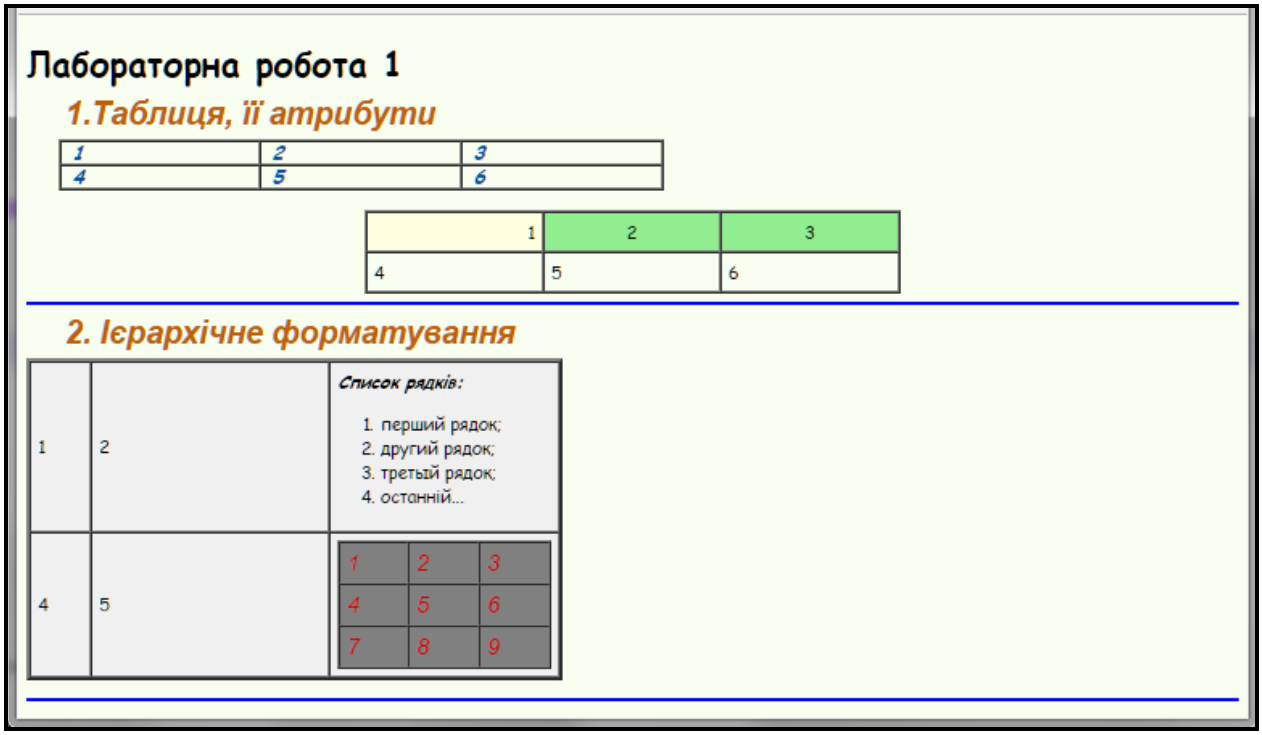


Рис.8

Контрольні запитання

1. *У чому відмінність використання у таблиці cellspacing та cellpadding?*
2. *Як зробити сітку таблиці невидимою?*
3. *Яким чином реалізувати "звичну" таблицю без відокремленості окремих комірок?*
4. *Яким чином можна зробити іншою зовнішню рамку таблиці?*
5. *Чому зараз прийнято форматування сторінки виконувати через CSS?*
6. *Яким чином винести у таблицю стилів форматування одного символу у параграфі?*
7. *У яких випадках таблицю стилів є сенс записувати окремим файлом, а коли включати у файл з контентом?*

Лабораторна робота №2.
Формування запитів від клієнта до сервера

Мета роботи: навчитись використовувати теги безпосереднього посилання та форми для створення запитів до web-сервера та вносити у запити дані, що передаються на сервер. Ознайомитись з основними елементами форми.

Загальні відомості

Прикладний застосунок передбачає виконання певних дій з обробки інформації, пов'язаних з елементами інтерфейсу користувача. Якщо розглядати такий застосунок в архітектурі web, ця обробка може бути виконана або на клієнтському боці (тобто браузером), або на сервері.

До відносно складної обробки інформації звернемось у наступних завданнях, а в даному випадку обмежимось рамками класичного web-ресурсу, який або є набором певної кількості пов'язаних між собою сторінок, або, як максимум усю обробку реалізує на сервері.

Перший варіант з двох зазначених є статичним web-сайтом, який фактично є незмінним багатосторінковим документом, присвяченим певній темі. Характерними його особливостями є:

- завантажуваність окремими статичними сторінками;
- змістовна пов'язаність сторінок;
- посилкова зв'язаність з опосередненим доступом з довільної сторінки на довільну іншу (в межах сайту).

Для реалізації треба забезпечити на кожній сторінці сайту елементи-посилання. Задачею посилань є активація у браузері відправки на сервер ідентифікації наступної сторінки. У відповідь сервер відправляє на бік

браузера потрібний html-файл з сукупністю всіх допоміжних даних, наприклад зображень.

Одразу зауважимо, що крім змістової пов'язаності є бажаним забезпечити й стилістичну. Забезпечити це зручно використанням відокремлених css-файлів, які є спільними для багатьох сторінок.

Наступним ускладненням роботи "web-класики" є відправка на сервер деяких нових даних, які, наприклад, повинні бути внесені у базу даних (як новий запис), або використовуються при виборі наступної сторінки. Для цього може бути застосована web-форма. За її допомогою формуються дані запиту з вказуванням серверної програми-обробника (традиційно разом з методом доступу закріплюється за формою) та спеціальним чином форматований рядок даних для цього обробника. При виконанні завдання даної лабораторної роботи використовується метод GET, завдяки чому легко побачити можливі помилки при формуванні такого рядка. Зауважимо, що обробку на сервері будемо виконувати дещо пізніше.

Завдання

1. *Реалізувати за зразком зовнішнього вигляду web-сторінку, з якої забезпечується виклик інших сторінок (статичний сайт).*
2. *Реалізувати за зразком зовнішнього вигляду форму.*

Хід виконання

1. Реалізація web-сторінок статичного сайту.

1.1. Реалізувати три сторінки, що викликаються, за базовим зразком, рис.9.

1.2. Реалізувати web-сторінку, з якої викликаються інші (зразок на рис.10). Увага; на сторінках, що викликаються, повинне бути повернення на головну сторінку.

2. Реалізація форми.

2.1. Створити документ з формою та одним елементом керування. Добитись правильного формування запиту з даними від цього елемента (для того, щоб бачити зміст запиту використати метод GET).

Рис.9

Рис.10

2.2. Доробити структуру форми до наведеного у зразку на рис.11.

2.3. Добитись правильного формування рядка запиту від усіх елементів форми.

Рис.11

Контрольні запитання

1. *Яким чином можна реалізувати посилання через натискання мишею зображення?*
2. *Що буде, якщо не задати ім'я елемента форми?*
3. *Чи є сенс робити на web-сторінці декілька форм?*
4. *У чому відмінність використання placeholder в порівнянні з використанням попереднього завдання значення поля введення тексту?*

Лабораторна робота №3. Реалізація графічного інтерфейсу користувача на основі використання HTML та JavaScript

Мета роботи: навчитись реалізовувати графічний інтерфейс користувача використанням HTML (разом з CSS), забезпечуючи інтерактивність використанням JavaScript.

Загальні відомості

Графічний інтерфейс користувача реалізується на основі зв'язування з певними подіями відповідних функцій-обробників. Основна перевага такої архітектури – забезпечення довільного порядку завдання програмі команд від користувача. Завдяки цьому, наприклад, є можливість у текстовому редакторі у довільний момент перейти від послідовного дописування тексту до редагування деякої його частини. Такий довільний порядок дій дозволяє також реалізувати взаємодію користувача з прикладною програмою аналогічно роботі з деяким пристроєм із певним набором елементів керування (кнопок, перемикачів, ручок, що обертаються, тощо).

У комп'ютерних технологіях для позначення чогось, що відбулось ззовні по відношенню до даної програми використовується термін "подія". Повідомлення про усі події (наприклад, натискання користувачем деякої клавіші, рух мишки тощо), які стосуються даної програми, надаються цій програмі. При цьому реалізується зовнішній виклик відповідних її функцій, які і виконують необхідні пов'язані з подією дії. Це трохи нагадує програмну обробку переривань.

При проектуванні довільного радіоелектронного пристрою зазвичай окремо дизайнером розробляється його корпус з елементами керування, до яких приєднуються розроблені інженером схемні реалізації функціональності пристрою. Аналогічно при проектуванні прикладного застосунку треба виконати:

1. розробку зовнішнього вигляду застосунку;
2. сукупність функцій-обробників.

В даному завданні використовуємо web-архітектуру. Таким чином перша з задач реалізується написанням web-сторінки застосунку за допомогою HTML та CSS. Зрозуміло, що в цьому випадку ми використовуємо тільки певну "декларацію", визначаючи як і де на сторінці повинні знаходитись певні елементи GUI. Реалізацію їх графічного вигляду повинна виконувати система виконання, роль якої в даному випадку бере на себе браузер. Для більшості випадків повністю вистачає стандартного набору елементів, що підтримується браузерами для вже знайомої за попереднім завданням web-форми.

Опис деякої нескладної обробки подій реалізується у "чистому" HTML. Це, наприклад, дозволяє за натисканням кнопки миші при певному положенні курсору на сторінці перемикати відтворюване браузером зображення, що, до речі, може бути використане для збільшення графічної привабливості застосунку заміною зображення кнопки на нестандартне.

Але для більш складної реакції web-застосунку на подію такого обмеженого набору готових дій системи обробки (браузера) недостатньо. Вирішити цю проблему дозволяє запуск браузером деякого специфічного для даного застосунку коду-обробника. Зрозуміло, що для його реалізації повинна використовуватись деяка мова програмування.

Формально для цього може бути використана довільна мова з відповідною системою виконання, або навіть компільована програма, виконання якої забезпечує операційна система. Але у справу втручається питання безпечності виконання такої програми. Заради цього доводиться накладати значні обмеження на дії, які може виконувати допоміжна програма, що запускається браузером, оскільки на комп'ютері користувача виконуються дії, завдання на які надходить ззовні. Наприклад не є припустимим безконтрольне втручання у файлову систему, зовнішні запити до неконтрольованих хостів, тощо.

Для забезпечення контрольованості такої програми-обробника подій слушно використати систему виконання, вбудовану у браузер, що відповідає технології виконання сценарію (скрипта). Зараз майже монопольно для такої потреби використовується така мова сценаріїв, як JavaScript.

Завдання

1. *Реалізувати інтерфейс користувача простого калькулятора.*
2. *Реалізувати функціональність кнопок калькулятора.*

Хід виконання

1. Реалізація інтерфейсу.

1.1. Розпочати новий документ Notepad++, записати його у форматі html.

1.2. Реалізувати необхідне розташування елементами керування калькулятора (рис.12). Для цього зручно використати таблицю, хоча можливі й інші способи форматування.

1.3. Реалізувати таблицю стилів, для відображення елементів калькулятора за зразком рис.12.

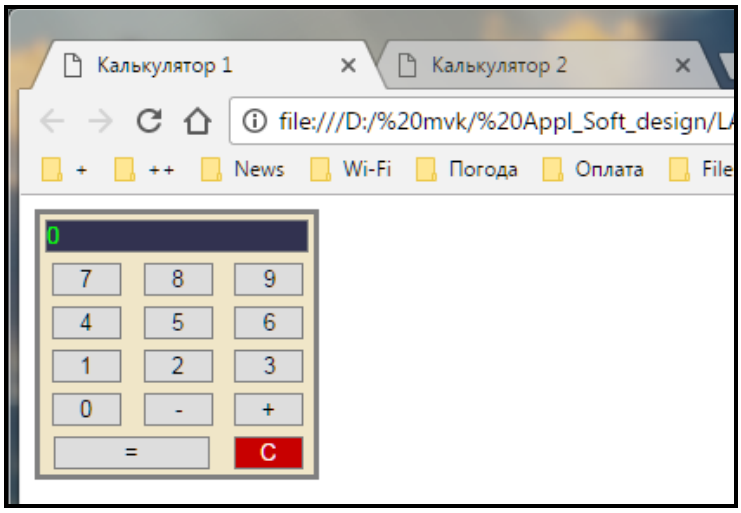


Рис.12

2. Реалізація функціональності.

2.1. Додати та реалізувати функції-обробники цифрових кнопок. Обробник повинен відповідну до кнопки цифру додавати до значення «поточного операнда» (того, що зараз набирається), помноженого на 10. Значення «поточного операнда» треба вивести на індикатор.

2.2. Додати та реалізувати функції-обробники кнопок арифметичних дій (« + » та « - »). Обробник повинен переписати поточний операнд в іншу змінну («попередній операнд») та обнулити «поточний операнд». Крім цього в змінну «дія» заноситься умовний код дії (деяке число). Вивести значення «поточного операнда» на індикатор.

Увага! Це завдання є навчальним і обробка даних є також частиною навчання. Відповідно реалізацію функцій-обробників для кнопок "калькулятора" треба робити саме через зміни тексту окремих операндів та запам'ятовування операцій, хоча, звичайно, простіше сформувати весь рядок дії з подальшим викликом функції eval.

2.3. Додати та реалізувати функцію-обробник кнопки-результату (« = »). Обробник повинен виконати відповідну до записаного коду дію з набраними операндами. Результат треба записати в «поточний операнд» та вивести його на індикатор.

2.4. Додати та реалізувати функцію-обробник кнопки «С».

2.5. Додаткове завдання. Максимально наблизити вигляд та функціональність калькулятора до зразка рис.13.

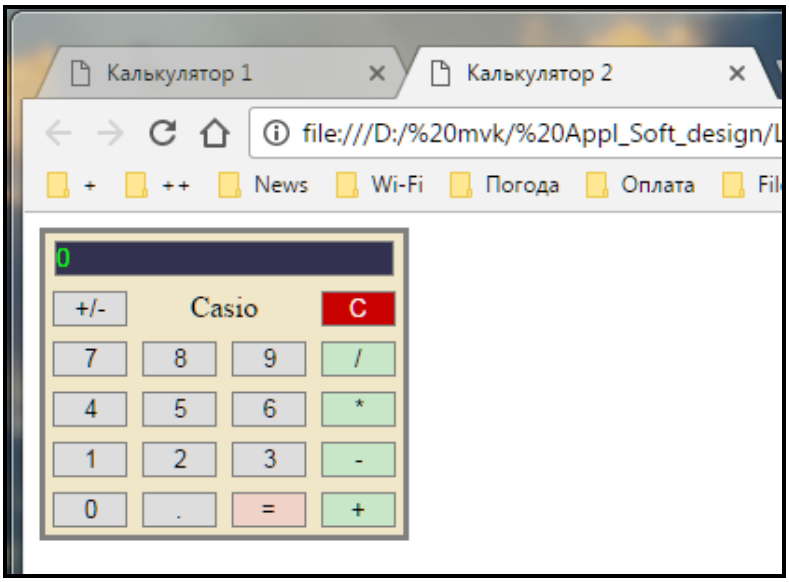


Рис.13

Контрольні запитання

1. *Які переваги для даного завдання надає використання CSS замість форматування атрибутами тегів?*
2. *У чому переваги та недоліки форматування таблицею в порівнянні з іншими способами?*
3. *Які є інші способи завдання положення кнопок калькулятора замість таблиці?*
4. *Яким чином реалізувати набір операндів простіше: відповідно завданню, або дописуванням цифри у рядок?*

Лабораторна робота №4. Інтерактивна зміна інтерфейсу користувача на основі використання HTML разом з JavaScript

Мета роботи: навчитись модифікувати HTML-сторінку, що відображається, за допомогою JavaScript без звернення до сервера.

Загальні відомості

Інтерактивність – термін, що означає таку організацію системи, коли потрібний результат досягається певною взаємодією, тобто інформаційним обміном, між її елементами. У багатьох випадках є бажаним забезпечувати зміну вигляду інтерфейсу користувача в залежності від потреб конкретної людини, яка працює на комп'ютері, його дій, конкретного завдання. Більш того, навіть окремі фази виконання завдання вимагають такого переналаштування.

Для забезпечення інтерактивності можуть бути також використані функції-обробники подій, за рахунок чого користувач не тільки може змінювати послідовність виконання, а й забезпечувати відображення або приховування певних даних або елементів керування. Цього можна досягти як при реалізації "десктопної" програми, так і web-застосунку.

У випадку web-архітектури застосунку для цього найбільш простим, що спадає на думку, є завантаження іншої сторінки. Але замість цього можна використати деяку програму (скрипт), яка взагалі не потребує звернення до сервера. Виконуваний на браузері скрипт дозволяє не тільки реалізувати обробники елементарних дій при реалізації GUI, а й реалізувати складні елементи керування, не передбачені у браузерній підсистемі відтворення стандартного набору елементів форми. Як приклад таких елементів можна навести ієрархічне спливаюче меню, календар тощо. Для цього можуть

навіть використовуватись функції малювання графічних об'єктів, які є у JavaScript (в межах даного курсу робота з цими функціями не розглядається).

Скрипти також дозволяють певним чином змінювати структуру та зовнішній вигляд відтворення даних на сторінці, що є основою, наприклад, "односторінкового" документа. У такому документі в певний момент візуалізується тільки та частина отриманих браузером даних, яка є необхідною саме на цій фазі роботи. Це дозволяє значно зменшити кількість звернень до сервера, тобто вилучає паузи, що виникають при перезавантаженні і відмальовуванні браузером нової сторінки. Реалізація "односторінкового" документа вимагає передачі на клієнтський бік певного набору даних, який не повинен бути відтворений на сторінці. З "класичного" HTML для такої потреби можна згадати приховане поле (атрибут hidden). Але цей спосіб приховування є не дуже зручним, особливо у випадку даних великого об'єму або складної структури. Значно кращим способом є запис цих даних у вигляді JavaSvript змінних, в тому числі масивів. Для візуалізації при цьому повинна виконуватись функція, яка дописує, або змінює певні HTML-теги.

Часткове відтворення даних браузером може використовуватись, наприклад, у випадках, коли сторінка є результатом відтворення результатів запиту до інформаційної системи. Сама таблиця, яка є інструментом відтворення цих даних, може формуватись як набір зрозумілих для браузера HTML-тегів або на сервері (будемо розглядати пізніше), або скриптом на клієнті. До речі, в цьому випадку на сторінці є доцільним додати певні елементи керування, які перебудовують таблицю внесенням у неї тільки частини отриманих з сервера даних, які необхідні саме у поточний момент.

Завдання

1. *Реалізувати за зразком зовнішнього вигляду HTML-сторінку для відображення даних, переданих з сервера.*

2. Реалізувати функціональність зазначеної сторінки.

Хід виконання

1. Реалізація HTML-сторінки

1.1. Завантажити прототип сторінки в Notepad++, ознайомитись з особливостями його реалізації.

1.2. Дописати проект для відтворення вигляду сторінки, показаного на рис.14.

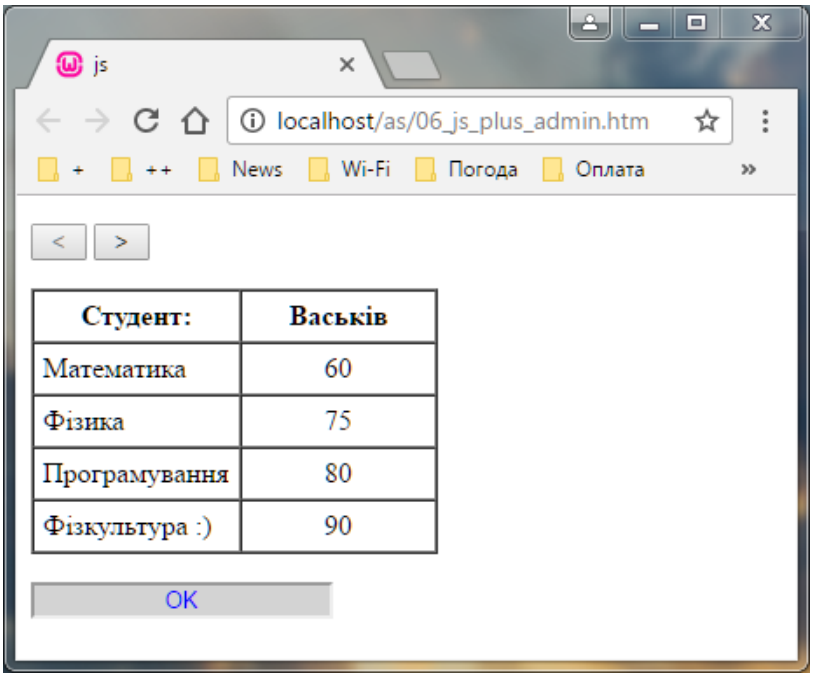


Рис.14

2. Реалізація функціональності.

2.1. Додати та реалізувати функції-обробники кнопок. Кнопки повинні перегортати дані з одного студента на іншого. При досягненні границі перегортання відповідна кнопка повинна ставати неактивною.

2.2. Додати аналіз оцінок студента, дані якого відображаються. Результат аналізу виводиться у текстове вікно. Якщо усі бали більше за 89 – відмінник (рис.15.а), якщо є бали нижче за 60

– є ризик відрахування (рис.15.б), якщо ж усе менше за 60 – відрахувати (рис.15.в). Інше – ОК (рис.14).

Студент:	Дорошенко
Математика	100
Фізика	98
Програмування	99
Фізкультура :)	90

Відмінник

а

Студент:	Рябокляч
Математика	10
Фізика	15
Програмування	25
Фізкультура :)	61

Ризик відрахування

б

Студент:	Петренко
Математика	50
Фізика	55
Програмування	30
Фізкультура :)	0

Відрахувати !

в

Рис.15

Контрольні запитання

1. *Яким чином за допомогою JavaScript забезпечується заміна текстового змісту певного HTML-тега?*
2. *Яким чином за допомогою JavaScript можна замінити стиль тега?*
3. *Як забезпечити виконання деякої функції одразу після завантаження web-сторінки?*
4. *Чи можна за допомогою JavaScript записати дані на комп'ютер користувача?*

Лабораторна робота №5. Інтерактивна зміна інтерфейсу користувача на основі запиту до сервера

Мета роботи: навчитись модифікувати web-сторінку **на боці серверу** на основі обробки отриманого від клієнта запиту за допомогою PHP-скрипта.

Загальні відомості

Зазвичай довільну задачу з обробки інформації можна розділити на певну сукупність підзадач, значною мірою відокремлених одна від одної, тобто кожна підзадача виконує майже самостійну обробку. Обмін інформації між ними зводиться до того, що результат роботи (вихідні дані) однієї з них використовуються як вхідні дані для іншої.

У випадку "десктопного" застосунку усе виконується на одному комп'ютері, але навіть у цьому випадку обробку даних можна розкласти на декілька потоків або процесів, що збільшить загальну швидкодію. Зрозуміло, що таке збільшення буде забезпечуватись тільки при умові багатоядерної архітектури комп'ютера.

Значно цікавішою є ситуація для web-застосунку. У цьому випадку частину дій можна перекласти на сервер, тобто використати клієнт-серверну архітектуру, яка вже згадувалась раніше.

Розподіл виконання між клієнтом (комп'ютером користувача) та сервером не завжди є однозначним. Звичайно є задачі які завжди краще виконувати на клієнті. До таких задач можна віднести перемальовування вікна, оскільки передача його повного растру мережею призводить до помітного збільшення мережевого трафіку та певної затримки відтворення змін для користувача.

Інші задачі завжди відносяться до сервера, як спільного ресурсу. До таких задач відносяться накопичення інформації для спільної роботи багатьох користувачів, забезпечення цілісності та синхронізація даних, деякі питання забезпечення безпеки тощо.

Щодо інших підзадач, при проектуванні клієнт-серверного застосунку при віднесенні їх до клієнта або сервера треба виходити з навантаженості завданнями кожного з боків системи, середньої та пікової кількості приєднань до серверу за одиницю часу і деяких інших факторів.

У даному завданні пропонується у повному обсязі повторити функціональність обробки даних, реалізовану у попередній роботі практикуму, але повністю перенести виконання з боку клієнта на бік сервера. Тобто з боку клієнта при обробці подій, пов'язаних з елементам GUI, повинні формуватись тільки короткі запити, а на серверному боці аналізується запит та виконуються усі дії, які раніше виконувались на браузері та формується тегова структура таблиці у форматі html.

Для реалізації серверної обробки даних використовується своя система виконання, пов'язана з web-сервером. В даному випадку - на основі використання скриптової мови PHP.

Одразу зауважимо, для цієї конкретної сукупності дій, передбачених завданням лабораторної роботи, оптимальним є виконання дій саме на клієнті, а перенесення на сервер носить виключно навчальний характер з метою навчитись виконувати певне завдання як на клієнті, так і на сервері.

Завдання

1. *Реалізувати за зразком зовнішнього вигляду динамічну web-сторінку для відображення даних з сервера.*
2. *Реалізувати функціональність зазначеної сторінки на основі обробки на серверному боці запиту від клієнта.*

Хід виконання

1. Реалізація динамічної web-сторінки.

1.1. Завантажити прототип сторінки в Notepad++, ознайомитись з особливостями його реалізації

1.2. Дописати проект відповідно до вигляду сторінки, аналогічному до попередньої лабораторної роботи (рис.14). Формування виду сторінки має бути реалізоване повністю **на боці серверу**.

2. Реалізація функціональності.

2.1. Додати та реалізувати обробку запиту від клієнта, який формується кнопками перегортання. Кнопки повинні перегортати дані з одного студента на іншого. При досягненні границі перегортання відповідна кнопка повинна ставати неактивною.

2.2. Додати аналогічний до попередньої лабораторної роботи аналіз **на боці серверу** оцінок студента, дані якого відображаються. Результат аналізу виводиться у текстове вікно. Нагадаємо: якщо усі бали більше за 89 – відмінник (рис.15.а), якщо є бали нижче за 60 – є ризик відрахування (рис.15.б), якщо ж усе менше за 60 – відрахувати (рис.15.в), інше – ОК (рис.14).

Контрольні запитання

1. *Де і чому краще виконувати функціональність даного завдання: на боці сервера чи на боці клієнта?*
2. *Чому засобами відладки браузера не можна перевірити виконання застосунку з цього завдання?*
3. *Яким чином можна на боці серверу реалізувати функціональність калькулятора з завдання №3, і чи є в цьому сенс?*

Лабораторна робота №6. Ускладнений варіант інтерактивної зміни інтерфейсу користувача на основі запиту до сервера

Мета роботи: навчитись модифікувати web-сторінку на боці серверу за допомогою PHP-скрипта на основі отриманого від клієнта запиту складної структури з даними для занесення до бази серверу.

Загальні відомості

У попередніх двох завданнях вважалось, що на web-сервері вже є деяка сукупність даних, отриманих від інформаційної системи. Це справедливо для виведення даних з інформаційної системи, але щоб вони там були їх треба попередньо ввести ззовні. Крім того вони можуть змінюватись у часі також за рахунок деяких зовнішніх дій. Для цього використовуються допоміжні програмні засоби. У випадку розподіленої системи функцію відправки в інформаційну систему нового блоку даних або редагування раніше записаних даних треба виносити на бік клієнта. Саме це і забезпечить властивість "розподіленості" системи, тобто територіальне рознесення окремих її складових.

У багатьох задачах, які є потрібними для забезпечення деякої спільної роботи багатьох людей, відправляти певні дані в інформаційну систему доводиться ручним введенням. Традиційним інструментом для цього є форма. До речі, сама ідея форми з'явилась задовго до появи web-технологій.

Фактично форма є бланком деякого документа, у якому певні поля, у які заноситься їх значення, мають своє фіксоване призначення для подальшого використання. Таким чином формалізацією даних форми є список, елементи якого є об'єднанням у пару ключа (ідентифікатора поля) та значення. Якщо згадаєте, яким чином будується сукупність даних web-форми

(лабораторна робота №2), стає зрозумілою аналогія при переході на web. У цій лабораторній роботі вже використовували web-форму для формували подібного списку. Але результатом був тільки сам рядок даних, який можна було побачити у браузері завдяки тому, що для відправки запиту на сервер використовувався метод GET. Самого ж обробника тоді не було.

У лабораторній роботі № 4 було реалізоване відтворення у вигляді таблиці даних, нібито отриманих з бази даних з можливістю перегортання записів на клієнті. Спробуємо до цього проекту додати обробку даних, які надходять на сервер із заповненої користувачем форми.

Таким чином, в даному випадку основна увага приділяється реалізації обробки на серверному боці даних запиту, які web-сервер надає програмі обробнику у вигляді деякого суперглобального масиву. У цьому випадку можна використовувати як метод GET, так і PUT. Нагадаємо, що ім'я програми обробника також задається у запиті від клієнта і визначається в описі тегу форми.

В рамках даного завдання обмежуємось дворівневою взаємодію (клієнт та сервер), не враховуючи потреби внесення отриманих від клієнта нових даних в інформаційну систему, тобто на сервер бази даних. Таке завдання (як подальший розвиток цього ж завдання) будемо виконувати трохи пізніше, оскільки перед цим треба навчитись звертатись до серверу бази даних - цьому буде присвячена наступна лабораторна робота.

Завдання

1. *Реалізувати за зразком зовнішнього вигляду форму, яка готує нові дані для передачі на сервер.*
2. *Реалізувати обробку даних цього запиту з внесенням їх у web-сторінку.*

Хід виконання

1. Реалізація web-сторінки з формою.

1.1. Завантажити прототип сторінки в Notepad++, ознайомитись з особливостями його реалізації.

1.2. Дописати проект відповідно до вигляду сторінки (рис.16). Формування виду сторінки повністю відбувається **на боці серверу**.

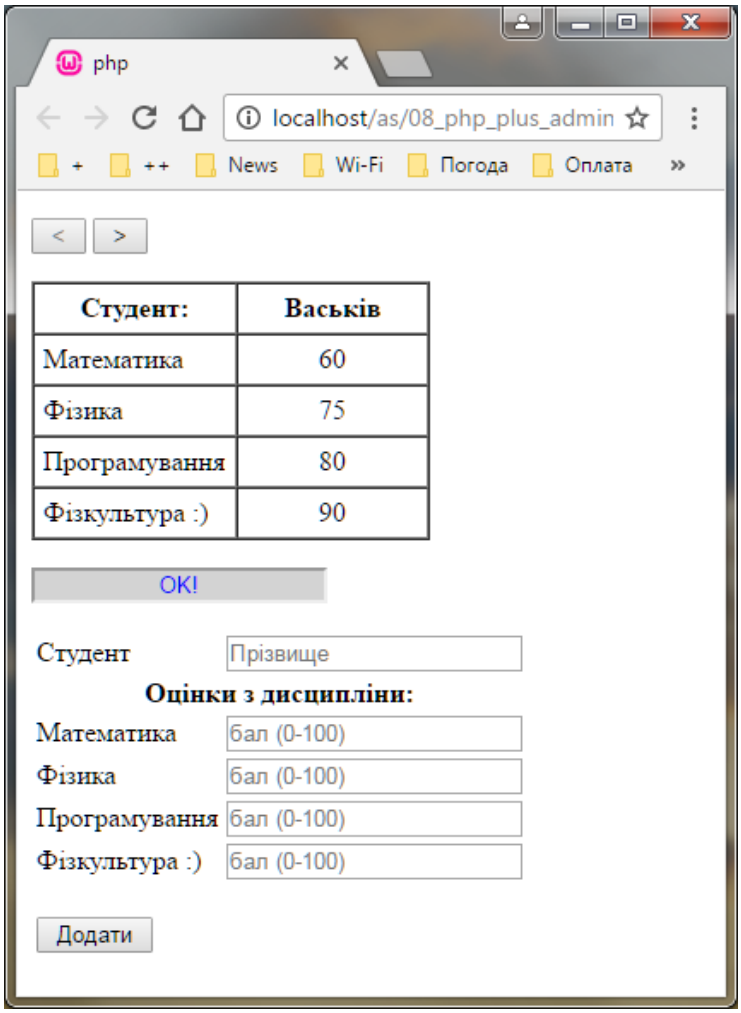


Рис.16

2. Реалізація функціональності серверного боку.

2.1. Додати та реалізувати обробку запиту від клієнта, який формується кнопкою «Додати». Дані повинні передаватись на сервер і приєднуватись до попередньо заданого у

завданні масиву даних, (який раніше можна було переглядати перегортанням).

2.2. Новий блок даних повинен відтворюватись як «активний» з продовженням перегортанням в межах зазначеного масиву даних (тобто не повинен блокувати перегортання).

2.3. Зауважимо, що після першого ж перегортання додані дані будуть стиратись, оскільки реальний запис до бази в даному навчальному завданні не реалізується, тобто при кожному запиті на сервер сукупність даних складається з прописаних явно і доданого блоку, якщо він є.

2.4. Додати (якщо це не було реалізовано у попередній роботі) аналіз **на боці серверу** оцінок студента, дані якого відображаються. Результат аналізу виводиться у текстове вікно повністю аналогічно до двох попередніх лабораторних робіт, тобто якщо усі бали більше за 89 – відмінник (рис.15.а), якщо є бали нижче за 60 – є ризик відрахування (рис.15.б), якщо ж усе менше за 60 – відрахувати (рис.15.с), інше – ОК (рис.14).

Контрольні запитання

1. *Чому доданий рядок з новими даними (в рамках функціональності даного завдання) при перегортанні втрачається?*
2. *Чим відрізняється отримання даних серверною програмою-обробником, при використанні методів PUT та GET?*
3. *Як блокувати можливі помилки роботи серверного обробника, якщо якесь поле не містить даних?*
4. *Чи можна для обробки даних на сервері використати не скрипт, а компільовану (наприклад, C++) програму?*

Лабораторна робота №7. Основні операції з базою даних

Мета роботи: навчитись створювати багатотабличну базу даних, вносити та змінювати дані, використовуючи SQL-запити

Загальні відомості

Сучасні комп'ютерні технології вимагають роботи з динамічно змінюваними даними великих обсягів. Для забезпечення високої ефективності такої роботи вони структуруються, у більшості випадків до табличного вигляду, тобто як реляційна база даних (БД). Для взаємодії програми з БД використовується спеціальна мова запитів SQL, за допомогою якої можна звертатись до відповідного програмного засобу – SQL-сервера. До сервера БД є можливість звертатись при використанні практично усіх сучасних мов програмування. Виключенням є JavaScript, і то у випадку, коли системою виконання є браузер (є варіант серверного використання цієї мови із значно ширшими правами).

Ієрархія доступу інформації на довільному SQL-сервері включає такі рівні:

1. бази даних;
2. таблиці в межах бази;
3. записи (рядки) в межах таблиці;
4. поля (колонки таблиці) в межах запису.

З врахуванням такої ієрархії загальна схема сеансу роботи з сервером БД включає такі кроки:

1. відкриття з'єднання з певним сервером від імені користувача (ідентифікується логіном) та паролю;
2. відкриття певної бази (ідентифікується ім'ям бази);
3. для даної бази задається запит на певні дії з таблицею (або декількома таблицями), ідентифікуються відповідними іменами;

4. сервер повертає результат у вигляді двозв'язного списку, з якого можна отримати записи і в межах запису отримати значення полів (деякі запити повертають скалярний результат) – цей пункт може повторюватись довільну кількість разів;
5. завершення з'єднання, відкритого на початку сеансу.

Відповідно для допуску до сервера необхідно мати логін та пароль, які задаються через адміністратора (для базового настроювання WampServer – "root" без пароля). Попередньо у вас не буде жодної бази з таблицями, тобто першим етапом роботи є створення бази (відповідним SQL-запитом) та створення таблиць необхідної будови в попередньо відкритій базі. Усі ці дії виконуються за описаним вище порядком.

Усі завдання виконуються на основі показаного на лекції допоміжного класу, який використовується для формування запиту до SQL-сервера та відтворення його результату у вигляді таблиці для перегляду у браузері (PHP-скрипт у файлі q.php, що надається до завдання). Логін та пароль для приєднання до серверу (при поточному, навчальному налаштуванні) у цьому скрипті прописані.

Для роботи з ним необхідно для кожного з розділів завдання записати у власну папку (на сервері) PHP-скрипт за зразком (рис.17). У цю ж папку слід записати сам файл q.php.

Нагадаємо, що сам текст SQL-запиту задається як поле функції q().

Щоб запобігти «перетину» даних з іншими студентами кожен студент має створити власну базу даних із іменем db_xx, де xx – номер варіанту, тобто умовний номер комп’ютера, на якому виконується завдання.

```
1
2  <?php
3  require_once "q.php";
4
5  //++++++++++++++++++++++
6  $db = 'db_01';  // !!!!!
7  //++++++++++++++++++++++
8
9  q("drop database $db");
10
11 ?>
```

Рис.17

Завдання

1. *Написати SQL-запити, які реалізують базу-каталог письмових робіт студентів навчального закладу.*
2. *Написати SQL-запити для заповнення реалізованої бази деяким набором даних.*
3. *Написати SQL-запити відкриття та модифікації даних в базі.*

Хід виконання

1. *Реалізація БД (типи полів та їх умовні назви – на розсуд виконавця).*

1.1. Створити базу даних з іменем db_xx, де xx – умовний номер комп'ютера, на якому виконується лабораторна робота.

1.2. Створити таблицю для даних про студентів з полями, які відтворюють:

- первинний ключ;
- прізвище;
- ім'я;
- курс;
- групу;

- рік зарахування на навчання.

1.3. Створити таблицю для даних про викладачів з полями, які відтворюють:

- первинний ключ;
- прізвище;
- ім’я;
- кафедру;
- посаду.

1.4. Створити таблицю для даних про дисципліни з полями, які відтворюють:

- первинний ключ;
- назву;
- кількість лекційних годин;
- кількість лабораторних годин;
- кількість семінарів;
- кількість модулів;
- вид фінального контролю (іспит або залік).

1.5. Створити таблицю для запису «письмових робіт» з полями:

- первинний ключ;
- посилання на студента;
- посилання на дисципліну;
- посилання на викладача, що виставив оцінку;
- оцінка за 100-бальною шкалою;
- вид роботи (модульна, фінальна);
- номер модуля;
- шлях на файл, де зберігається робота.

1.6. Вивести у веб-браузер структуру створених таблиць.

2. *Заповнення БД тестовими записами (задавати дані з такими значеннями, які дозволять виконати п.3).*

2.1. Додати до бази «студенти» 5 записів.

2.2. Додати до бази «викладачі» 2 записи.

2.3. Додати до бази «дисципліни» 4 записи.

2.4. Додати до бази «письмові роботи» 10 записів з посиланням на різних студентів, викладачів та дисципліни.

2.5. Вивести у веб-браузер усі дані створених таблиць (без фільтрації).

3. Відкриття та модифікація даних.

3.1. Відкрити базу даних студентів певного курсу.

3.2. Відкрити базу письмових робіт з «відмінними» оцінками.

3.3. Відкрити базу письмових робіт з «задовільними» оцінками.

3.4. Відкрити базу даних, так, щоб відтворювались прізвища студентів, викладачів, що оцінили роботу, назва дисципліни та оцінка усіх робіт, оцінених на «незадовільно».

3.5. Замінити усі оцінки, нижчі за 75 балів на 75.

Контрольні запитання

1. *Скільки баз може бути на сервері?*
2. *Скільки таблиць може бути у базі?*
3. *Чи можна звертатись до SQL-сервера, розташованого на іншому хості?*
4. *Як виводити результат відкриття таблиці "посторінково" (наприклад по 5 рядків)?*
5. *Як при виконанні SQL-запиту з його результату визначити назви полів таблиці?*

Лабораторна робота №8. Реалізація web-застосунку трирівневої архітектури

Мета роботи: навчитись модифікувати web-сторінку на основі взаємодії з базою даних (як отримання, так і додавання даних) з узгодженим використанням JS (клієнт), PHP (логіка сервера), SQL (дані).

Загальні відомості

У лабораторній роботі №4 розглядалась реалізація відображення окремого блоку даних у вигляді таблиці та перегортанні блоків обробкою дій користувача **на клієнті, тобто без перезавантаження сторінки**. У цьому завданні виходили з умови, що потрібна сукупність інформації від деякої бази даних вже отримана і має текстове представлення у вигляді масивів JavaScript.

Зміна відтворюваних на клієнтському боці даних без зайвих запитів до серверу при невеликій виборці з бази є оптимальною, оскільки забезпечує мінімальні затримки. Більш того, коли результатом вибірки з таблиці є велика кількість записів, зазвичай використовують "посторінкове" виведення такого результату. І в цьому випадку на клієнті може бути організована інтерактивна робота користувача зі сторінкою. Для роботи ж з наступною сторінкою при цьому виконується окремий запит до SQL-сервера (зрозуміло, що опосередковано через запит до web-сервера), але з додаванням у нього наступного діапазону виведення.

У лабораторній роботі №6 розглядалось розширення цього проекту додаванням передачі на web-сервер даних нового запису з web-форми, яка заповнюється користувачем, і попередньої обробки цих даних на сервері. Таким чином була виконана підготовка до наступного розширення – повноцінної реалізації застосунку трирівневої архітектури:

1. шар клієнта (HTML, CSS, JavaScript – все виконується браузером на комп'ютері користувача);
2. шар логіки (web-сервер і PHP-обробник, що запускається web-сервером);
3. шар даних (SQL-сервер).

Саме таке розширення попереднього проекту і пропонується виконати у даному завданні. Тобто зараз треба використати досвід, отриманий при виконанні попередньої роботи (виконання SQL-запитів) і додатково реалізувати наступну функціональність:

1. додавання нового запису до таблиці бази даних;
2. отримання деякої вибірки з бази даних і її оформлення у вигляді масивів, як ще було виконано у прототипі для виконання завдання роботи №4.

Нагадаємо, що для додавання даних у таблицю попередньо треба мати цю базу даних на SQL-сервері, а в ній необхідну таблицю. Таким чином, перш ніж реалізовувати певні дії з базою даних, її треба спроектувати, тобто визначити щодо конкретної задачі структуру таблиці (або таблиць для більш складного випадку) та відповідним SQL-запитом реалізувати базу і таблицю (таблиці) на сервері.

Завдання

1. *Реалізувати (за зразком) HTML сторінку для відображення даних, отриманих з бази даних.*
2. *Реалізувати форму для введення нового запису до бази даних.*

Хід виконання

1. Реалізація відображення даних з таблиці БД

1.1. Завантажити в Notepad++ прототип, що формує сторінку, яка виглядає аналогічно лабораторній роботі №6 (рис.16), ознайомитись з особливостями його реалізації.

1.2. Створити нову базу даних іменем db_xx, де xx – умовний номер комп'ютера, на якому виконується лабораторна робота. (або використати власну базу з попереднього завдання).

1.3. Створити нову таблицю, що відповідає структурі даних, яка відповідає застосунку-прототипу.

1.4. Додати три тестові записи до таблиці для подальшої відладки функціональності застосунку.

1.5. Замінити тестові статичні дані у застосунку-прототипі на ті, що отримуються з бази даних (відкриття усієї таблиці без фільтрації).

2. Реалізація запису даних у таблицю БД

2.1. Доповнити систему відтворення даних з таблиці введенням даних «нового студента» з клієнтської форми. При цьому після підтвердження внесення на клієнті повинні одразу відтворюватись дані саме нового запису з можливістю перегортання усіх записів.

2.2. (Додаткове завдання!) Доповнити сторінку засобами редагування (із зміною даних в таблиці) поточного запису.

Контрольні запитання

1. *Яким чином треба модифікувати SQL-запит для посторінкового виведення його результату?*
2. *Яким чином можна для створення таблиці задати умову її відсутності?*
3. *Яким чином можна з'ясувати кількість (тільки) записів у результаті фільтрації таблиці без отримання повних даних такої фільтрації?*
4. *Чи може бути використаний SQL-сервер на іншому хості (не там, де працює web-сервер)?*

Лабораторна робота №9. Реалізація графічного інтерфейсу користувача «десктопного» застосунку

Мета роботи: навчитись реалізовувати графічний інтерфейс користувача на прикладі WPF-застосунку (на платформі MS.Net).

Загальні відомості

Є багато різних бібліотек для реалізації графічного інтерфейсу користувача (GUI). Одним з найбільш поширених засобів є Visual Studio від Microsoft. Для виконання даного завдання використовується саме цей засіб (версія – з 2010).

Нагадаємо, що розробка інтерфейсу користувача зводиться до:

1. реалізації вигляду GUI розміщенням та налаштуванням параметрів відображення усіх елементів GUI;
2. програмування функцій-обробників, пов'язаних з надходженням певних подій (входження курсору в область даного елемента, натискання кнопки, потреба перемальовування тощо).

Не зважаючи на графічний вигляд GUI результат виконання першого із зазначених завдань повинен бути все одно реалізований як деякий текст. В більшості засобів розробки це можна зробити в межах синтаксису мови програмування, з якою працює цей засіб. Іноді для цього використовується спеціальна окрема мова.

У MS.Net основною архітектурою віконного GUI прикладної програми досить довго була Windows Forms. В цьому випадку для запису результатів проектування GUI використовувався базовий синтаксис мови C#. Оскільки виключно текстове програмування GUI є досить складним і вимагає багато часу, було реалізовано спеціальний інструмент графічного програмування.

Редагування положення та розмірів елемента GUI виконується мишею. Зміна його параметрів відбувається використанням спеціальної форми властивостей. Але результат завдання елементів та усіх їх параметрів записується звичайним синтаксисом C#, правда, в окремому файлі. В принципі текст програми у цьому окремому файлі можна редагувати і прямим чином.

Зараз технологія Windows Forms вважається застарілою. Їй на заміну запропонована суттєво інша - WPF (Windows Presentation Foundation). У цьому випадку також можна реалізувати елементи GUI мовою C#, але "у штатному режимі" засіб графічного проектування інтерфейсу користувача записує результати роботи окремою мовою XAML (спеціальною модифікацією XML). Але загальні принципи графічного програмування та базові його інструменти залишились в основному незмінними. Загалом при графічному програмуванні виконуються такі кроки:

1. На спеціальній панелі мишею обирається елемент керування і переноситься на потрібне місце майбутнього вікна застосунку;
2. Мишею можна змінювати розміри і положення елемента;
3. У формі "Properties" редагуються параметри активованого мишею елемента;
4. Через закладку "Events" цієї ж форми обирається подія і кліком миші на рядку з обраною подією запускається автоматичне створення заготовки програми-обробника, функціональність якої дописується вже звичайним текстовим програмуванням.

В даному завданні пропонується використовувати актуальний спосіб розробки GUI, саме тому є обмеження версії Visual Studio з версії 2010, у яку було додано підтримку WPF.

Завданням даної роботи є реалізація функціональності калькулятора. Подібне завдання вже виконувалось в іншій роботі цього практикуму. Це дозволяє зрозуміти спільні риси подіє-орієнтованої архітектури прикладного застосунку при порівнянні використання різних технологій. Одразу відмітимо, що

алгоритм самих обробників у цих двох випадках може бути однаковим (хоч і не обов'язково), що при значній схожості мов програмування дозволяє перенести з невеликою модифікацією текст з іншого проекту.

При реалізації WPF-калькулятора слід брати до уваги, що функціональність обробників можуть бути реалізована безпосередньо у тілі цих функцій, але більш правильним підходом є написання "калькулятора" як окремого класу з викликом відповідних методів цього класу з функцій обробників елементів GUI.

Завдання

1. *Реалізувати за зразком зовнішнього вигляду інтерфейс простого калькулятора.*
2. *Реалізувати функціональність калькулятора.*

Хід виконання

1. Реалізація інтерфейсу

1.1. Розпочати новий проект типу WPF Application.

1.2. Додати потрібні елементи керування за зразком, наведеним на рис.18 (відповідає наданому для виконання лабораторної роботи Wpf_Calc.exe).

1.3. Виконати настроювання елементів керування, використовуючи форму властивостей та XAML (порівняйте, що зручніше).

2. Реалізація функціональності

2.1. Додати та реалізувати функції-обробники цифрових кнопок. Обробник повинен відповідну до кнопки цифру додавати до значення «поточного операнда» (того, що зараз набирається), помноженого на 10. Значення «поточного операнда» треба вивести на індикатор за зразком:

```
textBox1.Text = x0.ToString();
```

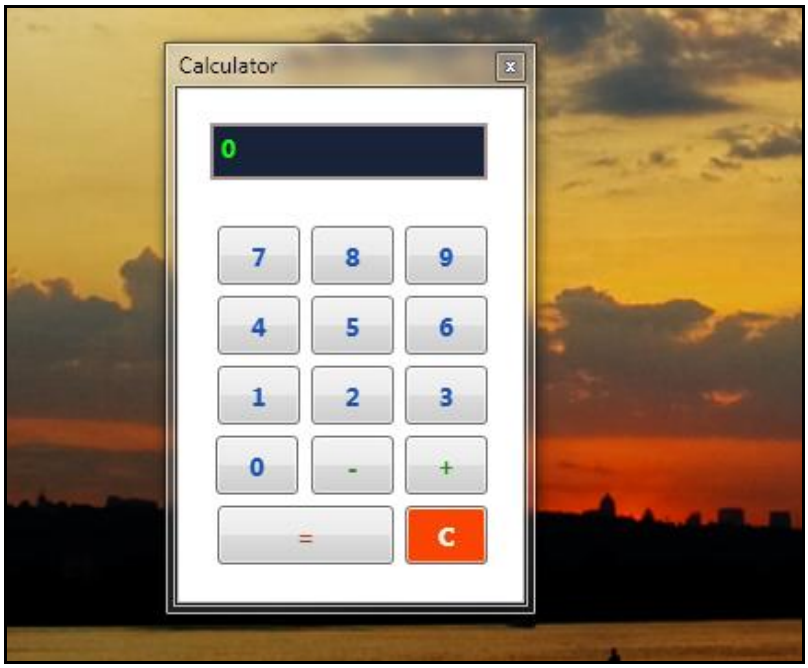


Рис.18

2.2. Додати та реалізувати функції-обробники кнопок-арифметичних дій (« + » та « - »). Обробник повинен переписати поточний операнд в іншу змінну («попередній операнд») та обнулити «поточний операнд». Крім цього в змінну «дія» заноситься умовний код дії (деяке число). Вивести значення «поточного операнда» на індикатор.

2.3. Додати та реалізувати функцію-обробник кнопки-результату (« = »). Обробник повинен виконати відповідну до записаного коду дію з набраними операндами. Результат треба записати в «поточний операнд» та вивести його на індикатор.

2.4. Додати та реалізувати функцію-обробник кнопки «С».

2.5. Додаткове завдання. Максимально наблизити вигляд та функціональність калькулятора до зразка Wpf_Calc_plus.exe (рис.19)

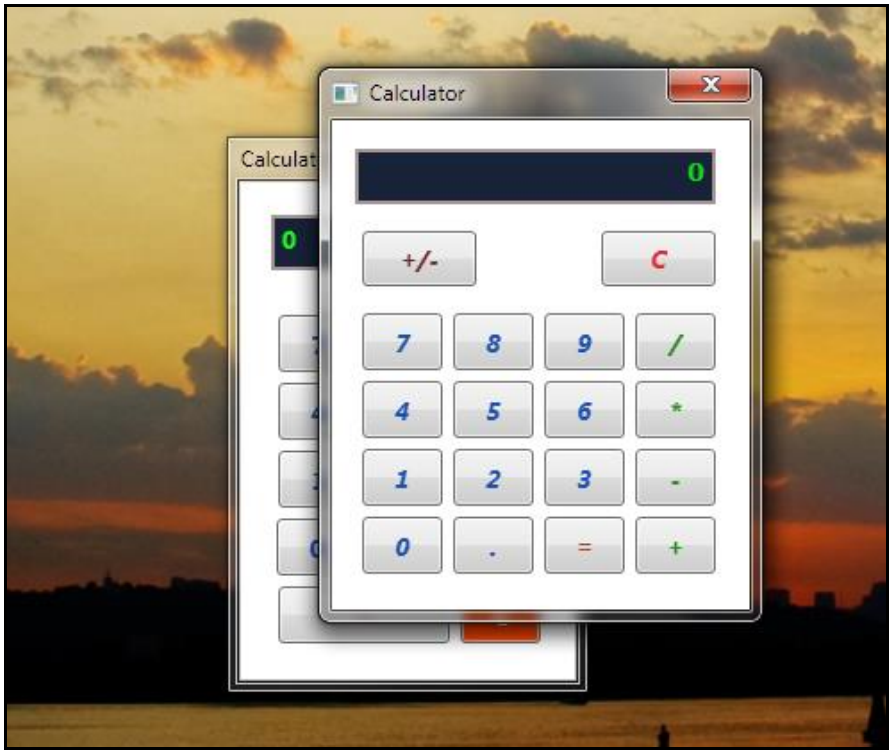


Рис.19

Контрольні запитання

1. *Чи можна замінювати назву функції-обробника?*
2. *Чи можна реалізувати окремі обробники на натискання кнопки миші та її відпускання?*
3. *Як реалізувати спільну функцію-обробник для декількох елементів?*
4. *Які є способи змінити розмір елемента GUI, наприклад кнопки?*
5. *Що є спільного в XAML та HTML?*

Література

1. *Н. Комолова, Е. Яковлєва HTML: Самоучитель. 2-е изд. — СПб.: Питер, 2011. — 288 с.*
2. *Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-е изд. — СПб.: Питер, 2016. — 768 с.*
3. *Прохоренок Н.А., Дронов В.А. HTML, JavaScript и MySQL. Джентльменский набор Web-мастера. – 4 изд. СПб.: "БХВ-Петербург", 2015. - 768с.*
4. *Колесниченко Д.Н. PHP и MySQL. Разработка веб-приложений. – 5 изд. СПб.: "БХВ-Петербург", 2015. – 592 с.*
5. *Український веб-довідник. https://css.in.ua/*
6. *Web-технологии. https://htmlweb.ru/*