ВСУП

Методичні рекомендації «Основи програмування у спорті» передбаченні для магістрів факультету фізичної культури, спорту та здоров'я. Вони також будуть корисними для початківців, які цікавляться програмуванням.

У цьому виданні ми стисло та доступно створимо уявлення про найбільш поширені мови програмування. Узагальнимо інформацію про макроси, яка дасть змогу зробити перші кроки до написання найпростіших програм. Ми також поверхнево ознайомимося з деяким інтегрованим середовищем для створення програм. Зокрема, навчимось писати найпростіші консольні програми та програми VimForms у середовищі Visual Studio. Вивчимо основи Web-програмування та роботи з базами даних.

Електронна версія видання супроводжується інтерактивними гіперпосиланнями на важливі терміни, визначення та поняття. Це дасть змогу перетворити навчання у цікавий та творчий процес. Натомість, паперова версія містить тлумачний термінологічний словник.

Інформація, систематизована в методичних рекомендаціях, розрахована на початківців. Це ознайомчі кроки у складній сфері ІТ-технологій. Для подальшого саморозвитку та професійного становлення необхідно систематично і багато вчитись.

Методичні рекомендації «Основи програмування у спорті» написані згідно з силабусом одноіменної вибіркової освітньої компонети навчального плану спеціальності 017 Фізична культура і спорт освітньо-професійної програми Фізична культура і спорт.

Ідея: створити електронну методичку з активними посиланнями, так само силабус з інтерактивними посиланнями, навчальний план!!!!

https://www.twirpx.link/files/applied/comp/

ідеї – в електронній версії залишити активними всі посилання. У бумажній зробити глосарій з термінологією та

Зміст

2. **Работа в среде Visual Studio .NET**

4. Робота з таймером, часом та датою

5. Обєктне програмування.

5. Програмування і інтернет.

6. Програмування і бази даних

МОВИ ПРОГРАУМУВАННЯ

Анотація. *Мова програмування – це штучна мова для передачі команд комп'ютерам. Кожна мова програмування має свою граматику, словник і синтаксис. Вони визначають вигляд програми і те, як пишуться змінні, оголошення та інші мовні конструкції. Мови програмування класифікуються за рівнем абстракції (високого та низького рівня), за ділянкою застосування (спеціалізовані та універсальні), за підтримуваною парадигмою (об'єктно-орієнтовані, логічні, функційні та структурні). Мови програмування можуть бути реалізовані як компільовані та інтегровані.*

Мова програмування (англ. *Programming language*) – це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам. Мови програмування використовуються для створення програм, які контролюють поведінку машин, та для запису алгоритмів.

З часу створення перших програмованих машин було створено понад дві з половиною тисячі мов програмування. Щороку до них додаються нові. Деякими мовами вміє користуватись тільки невелике число їхніх розробників, інші стають відомі мільйонам людей. Професійні програмісти зазвичай застосовують у своїй роботі декілька мов програмування.

Синтаксис мови програмування визначає те, як буде виглядати програма цією мовою, зокрема, як пишуться оператори, оголошення й інші мовні конструкції.

Область зберігання даних в апаратній частині комп'ютера (пам'ять, регістри та зовнішні запам'ятовувальні пристрої) зазвичай мають доволі просту структуру в вигляді послідовності бітів, згрупованих в байти або слова. Проте в віртуальному комп'ютері, як правило, організовано складнішим чином – в різні моменти виконання програми використовуються такі форми зберігання даних, як стеки, масиви, числа, символьні рядки та інші. Один або декілька однотипних елементів даних, об'єднаних в одне ціле в віртуальному комп'ютері в певний момент виконання програми, заведено називати *об'єктом даних*. При виконанні програми існує багато об'єктів даних різних типів.

Тип даних – це деякий клас об'єктів даних разом з набором операцій для створення і роботи з ним. В кожній мові програмування є певний набір вбудованих примітивних типів даних. Додатково в мові можуть бути передбачені засоби, що дають змогу програмістові визначати нові типи даних.

Мови класифікують за такими критеріями:

Рівень абстракції

Мови програмування високого рівня оперують сутностями зрозумілими людині, такими як об'єкти, змінні, функції. Мови програмування нижчого рівня оперують сутностями зрозумілими машині: байти, адреси, інструкції. Текст програми мовою високого рівня зазвичай набагато коротший ніж текст такої самої програми мовою низького рівня, проте програма має більший розмір.

Ділянка застосування

Універсальні та спеціалізовані. Спеціалізовані мови теж бувають Тюрінг-повні, та все ж їх ділянка застосування обмежена, як, наприклад, у мови shell.

Підтримувані парадигми програмування

Об'єктно-орієнтовані, логічні, функційні, структурні…

Мови програмування можуть бути реалізовані як компільовані та інтерпретовані.

Програма компільованою мовою за допомогою компілятора (особливої програми) (компілюється) в машинний код (набір інструкцій) для даного типу процесора, що записується у об'єктний модуль. З одного або кількох об'єктних файлів компонувальник формує виконуваний файл, який може бути запущений на виконання як окрема програма (прикладом є виконувальні файли з розширенням *.exe). Іншими словами, компілятор переводить вихідний текст програми з мови програмування високого рівня в двійкові коди інструкцій процесора.

Якщо програма написана скриптовою мовою, то інтерпретатор безпосередньо виконує (інтерпретує) вихідний текст без попереднього перекладу. При цьому програма залишається мовою оригіналу і не може бути запущена без інтерпретатора. Процесор комп'ютера, в зв'язку з цим, можна назвати інтерпретатором для машинного коду.

Підпрограми діляться на процедури та функції: Синтаксично процедури та функції складаються з *заголовка* та *тіла*.

Тіло процедури, як і програми, своєю чергою може містити описи процедур і функцій. Таким чином, процедури і функції можуть бути вкладені одна в одну як завгодно глибоко, при цьому тіло програми – саме верхнє в ланцюжку.

Процедури відрізняються від функцій тим, що функції повертають якесь значення, а процедури – ні.

Об'єктно-орієнтоване програмування (ООП) – це технологія створення складного програмного забезпечення, яке засноване на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію зі спадкоємством властивостей.

Основна перевага ООП – це значне спрощення процесів створення та модифікації програмних систем. Набагато легше маніпулювати 100 об'єктами, кожен з яких сам відповідає за свою поведінку і обробку даних пов'язаних з ним, ніж тисячами функцій розкиданих по різних модулях.

BASIC (*Бейсик*) – (від англ. *Basic* – початковий, елементарний) мова програмування високого рівня, метою створення якої було отримати просту в користуванні мову для початківців. Мова набула поширення у 1989-х, і лишається популярною й досі, маючи чимало діалектів. Мову було створено у 1963 професорами Дартмутського коледжу Джоном Кемені і Томасом Курцом.

Назва мови є акронімом від фрази англ. *Beginner's All-purpose Symbolic Instruction Code*, що означає «універсальний код символьних інструкцій для початківців».

Мова бейсик постійно розвивається. Вона суттєво оновлюється після створення корпорацією Microsoft програмно середовища Visual Basic (VB), яке дає змогу користувачам засобами візуального програмування складати власні програми для операційної системи Windows.

Для розширення можливостей таких програм, як MS Word, MS Excel, MS Access ті інших корпорація Microsft розробила версію VB під назвою Visual Basic for Applications (VBA).

Наступне удосконалення мови – Visual Basic. NET, що є складовою інтегрованого середовища створення програм Visual Studio .NET. Visual Basic .NET дає змогу не лише створювати типові навчальні програми, що можна робити і в середовищі Qbasic, чи типові Windows-проекти, для чого достатньо середовища Visual Basic, але й високопрофесійні програми для роботи в локальній мережі, а також в інтернеті з характерним web-інтерфейсом, що запускаються з сервера, доступ до яких здійснюється за допомогою браузера.

Алфавіт мови складається з таких символів:

- латинських літер від A до Z та літер кирилиці;
- цифр від 0 до 9;
- символів математичних операцій (+,-,*,/,^);
- символів відношення (=, <,> та інших);
- розділових символів (крапка, кома, двокрапка, крапка з комою, лапки, круглі дужки, пропуск);
- спеціальних символів (!,#,$,%,^,& та інших).

Алфавіти можуть несуттєво відрізнятися в різних версіях мови. За допомогою символів алфавіту будують елементи мови: службові слова, імена змінних, числа, вирази, імена, функції та інші.

Програми складається з команд. Іноді команди називаються операторами, вказівками, реченнями. За призначенням команди поділяють на такі групи:

- описові команди, які використовують для опису даних, типів змінних, розмірів масивів, нестандартних функцій тощо;
- команди введення-виведення даних, які використовують для введення виведення інформації, зокрема, для організації діалогу користувача з комп'ютером.
- команди для обчислень і керування процесами опрацювання інформації: команди надання значення, переходу, розгалуження, циклів тощо;
- інші команди, які забезпечують додаткові можливості: команди для роботи з файлами даних, команди для графічних побудов, отримання звукових ефектів тощо.

Команди складаються з неподільних елементів мови (літералів): службових слів, чисел, символів операції тощо.

Синтаксис містить типові ключові слова BASIC.

Python (найчастіше вживане прочитання – «**Пáйтон**», запозичено назву з британського шоу Монті Пайтон) – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидко ї розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та

стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Python портованна і працює майже на всіх відомих платформах – від КПК до мейнфреймів.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді.

Подібно Ліспу та Прологу в режимі відлагодження, інтерпретатор Python має інтерактивний режим роботи, при якому введені з клавіатури вирази відразу ж виконуються, а результат виводиться на екран. Цей режим цікавий не тільки новачкам, але й досвідченим програмістам, які можуть протестувати в інтерактивному режимі будь-який фрагмент коду, перш ніж використовувати його в основній програмі, або просто використовувати як калькулятор з великим набором функцій.

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною, потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими об'єктно-орієнтованими мовами.

Програмне забезпечення (застосунок або бібліотека) на Python оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в *пакунки*. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування. Наприклад, в стандартній бібліотеці є «чистий» модуль pickle і його аналог на Сі: cPickle. Модуль оформляється у вигляді окремого файлу, а пакет – у вигляді окремого каталогу. Підключення модуля до програми здійснюється оператором import. Після імпорту модуль представлений окремим об'єктом, що дає доступ до простору імен модуля. У ході виконання програми модуль можна перезавантажити функцією reload().

Python підтримує повну інтроспекцію часу виконання. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.

Багата стандартна бібліотека є однією з привабливостей мови Python. Тут є засоби для роботи з багатьма мережевими протоколами та форматами Інтернету, наприклад, модулі для написання HTTP-серверів та клієнтів, для розбору та створення поштових повідомлень, для роботи з XML, тощо. Набір модулів для роботи з операційною системою дозволяє писати крос-платформні застосунки. Існують модулі для роботи з регулярними виразами, текстовими кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, серіалізацією даних, юніт-тестуванням та ін.

Крім стандартної бібліотеки існує багато інших, що надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі Win32

підтримуються всі виклики Win32 API, а також COM в обсязі не меншому, ніж у Visual Basic або Delphi. Існує велика кількість прикладних бібліотек для Python у різноманітних галузях: веброзробка, бази даних, обробка зображень, обробка тексту, чисельні методи, програми операційної системи тощо.

З Python поставляється бібліотека tkinter на основі Tcl/Tk для створення крос-платформних програм з графічним інтерфейсом. Для науково-технічної мети найбільшого поширення набуло використання matplotlib – бібліотеки з інтерфейсом, аналогічним MATLAB Plot Tool. Для створення ігор та програм, що вимагають нестандартного інтерфейсу, можна використовувати бібліотеку Pygame. Вона також надає великі засоби роботи з мультимедіа: з її допомогою можна керувати звуком і зображеннями, відтворювати відео.

Java (вимовляється *Джава* – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Спочатку мова називалася Oak («дуб») і розроблялася Джеймсом Гослінгом для програмування побутових електронних пристроїв. Згодом вона була перейменована в Java і стала використовуватися для написання клієнтських застосунків і серверного програмного забезпечення. Названа на честь марки кави Java, яка, в свою чергу, отримала найменування однойменного острова (Ява), тому на офіційній емблемі мови зображена чашка з паркою кавою. Існує й інша версія походження назви мови, пов'язана з алюзією на каво-машину як приклад побутового устаткування, для програмування якого спочатку мова створювалася.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано.

JavaScript (JS) – динамічна, об'єктно-орієнтованаhttps://uk.wikipedia.org/wiki/JavaScript прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має кілька властивостей, притаманних функціональним мовам, – функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання (closures) – що додає мові додаткову гнучкість.

JavaScript має C-подібний синтаксис, але в порівнянні з мовою C має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого класу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичне збирання сміття;
- анонімні та стрілочні функції.

JavaScript містить декілька десятків вбудованих об'єктів, які поділяються на групи: фундаментальні (Object, Function, Boolean, Symbol), помилки (група об'єктів Error), числа та дати (Number, BigInt, Math Date), текстові (String, RegExp), індексовані (група об'єктів Array), ключові (Map, Set, WeakMap, WeakSet), для роботи з структурованими даними (ArrayBuffer, Atomics, DataView, JSON), абстрактні (Promise, Generator), рефлекційні (Reflect, Proxy), групи Intl та WebAssembly. Крім того, JavaScript містить набір вбудованих операцій, що керують логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений у порівнянні з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, наприклад, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може знаходитися в тексті програми після неї.

Семантика мови схожа з мовою Self.

При використанні в рамках технології DHTML JavaScript код включається в HTML-код сторінки і виконується інтерпретатором, вбудованим в браузер. Код JavaScript вставляється в теги <script></script>, хоча в більшості браузерів мова сценаріїв за умовчанням саме JavaScript.

При розробці великих і нетривіальних вебзастосунків з використанням JavaScript, критично важливим є доступ до інструментів відлагодження. Оскільки браузери, від різних виробників, дещо відрізняються у поведінці JavaScript і реалізації Об'єктної моделі документа, необхідно мати відлагоджувач для кожного браузера, якщо вебзастосунок орієнтовано на нього.

На даний час Firefox, Opera, Google Chrome, Edge та Safari мають зневаджувачі для себе.

HTML (англ. *HyperText Markup Language* – мова розмітки гіпертексту) – стандартизована мова розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені *тегами*, написаними з використанням кутових дужок. Теги на кшталт <img /> чи <input /> безпосередньо виводять

вміст на сторінку. Інші теги, такі як <p>, оточують текст і надають інформацію про нього, а також можуть включати інші теги як піделементи. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

В HTML можна вбудовувати програми, написані на скриптових мовах, наприклад JavaScript, які впливають на поведінку та вміст вебсторінок. Включення CSS визначає вигляд і компонування вмісту. World Wide Web Consortium (W3C), який супроводжує стандарти HTML та CSS, заохочує використання CSS над явним презентаційним HTML з 1997 року.

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації зі Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

Для поліпшення взаємодії SGML вимагає аби кожна похідна мова (HTML у тому числі) визначала свою кодову таблицю для кожного документа, яка складається з *репертуару* (перелік різноманітних символів) та *позиції символу* (перелік цифрових посилань на символи з репертуару). Кожен документ HTML – це послідовність символів із репертуару.

HTML використовує найповнішу кодову таблицю UCS (англ. *Universal Character Set* – Універсальний Набір Символів).

Проте однієї кодової таблиці недостатньо для того, щоб браузери могли правильно відтворювати документи HTML. Для цього браузерам потрібно «знати» специфічну кодову таблицю документа, яку автор має зазначати завжди в елементі meta із параметром charset. За замовчуванням використовується кодова таблиця ISO-8859-1, відома також як Latin-1.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

Документ HTML 5.2 складається з трьох частин:

1. **Декларація типу документа** (англ. *Document type declaration*, Doctype), на початку документа, в якій визначається тип документа (DTD).
2. **Шапка документа** (знаходиться в межах елемента head), в якій записано загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;
3. **Тіло документа** (може знаходитися в елементі body), в якому міститься основна інформація документа.

Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та вміст (контент). Існують певні настанови щодо кожного атрибута та контенту елемента, які треба виконувати задля того, щоб HTML-документ був визнаний валідним.

У елемента є початковий тег, який має вигляд <*element-name*>, та кінцевий тег, який має вигляд </*element-name*>. Атрибути елемента записуються в початковому тегу одразу після назви елемента, контент елемента записується між його двома тегами. Наприклад: <*element-name element-attribute="attribute-value*">контент елемента</*element-name*>.

Деякі елементи, наприклад br, не містять контенту, тож і не мають кінцевого тегу. Елемент може не мати початкового та кінцевого тегу (наприклад, елемент *head*), проте він завжди буде представлений в документі. Нижче зазначені деякі типи елементів розмітки HTML.

Елементи структурної розмітки застосовуються задля опису семантики тексту, іншими словами ці елементи описують призначення тексту свого контенту. Вони не зазначають ніякого спеціального (візуального) відтворення тексту, проте більшість браузерів мають стандартні стилі форматування для кожного елемента. Для подальшого стилізування тексту рекомендується використовувати Каскадні таблиці стилів (CSS).

Більшість з атрибутів елемента являє собою пару «назва-значення», розділених між собою знаком рівності, та записаних у початковому тегу одразу після назви елемента. Значення атрибуту може бути оточене лапками (подвійними або одиничними), також, якщо значення атрибуту складається з певних символів, його можна не виділяти лапками зліва. Проте невзяття значення атрибутів у лапки вважається небезпечним кодом. На відміну від атрибутів виду «назва-значення», є певні атрибути, що впливають на елемент, назва яких лише з'явилась в початковому тегу (наприклад, атрибут *ismap* елемента *img*).

Оскільки HTML є похідною мовою від SGML, усі типи даних HTML ґрунтуються на базових типах даних SGML (наприклад, *PCDATA*, *CDATA*, *NAME*, *ID*, *NUMBER*).

Кожен елемент має дві властивості – *атрибути* і *вміст*, які мають певні значення. Всі можливі значення цих двох властивостей прописуються відповідно до визначених у DTD типів даних.

Існують такі випадки, коли в документі потрібно використати якийсь символ, якого немає в обраній для документа кодовій таблиці. Для таких випадків можливо замінити символ на еквівалентне йому SGML-посилання на символ (мнемоніку).

Розрізняють мнемоніки двох видів:

- *Цифрові мнемоніки (десяткові або 16-кові).* Визначають кодову позицію символу із таблиці кодів UCS.
- *Мнемоніки із певних сполучень символів.* Такі мнемоніки використовують псевдоніми замість кодів символів. Проте в HTML не визначені псевдоніми для кожного символу із UCS.

Так само як і кожна мова, будь-яка комп'ютерна мова має свою власну граматику, словник і синтаксис. І кожен документ, написаний цією мовою, має дотримуватися цих правил. HTML використовує машинно-зчитуючу граматику, яка називається DTD, механізм, успадкований від SGML.

Проте, так само як і тексти природної мови можуть містити граматичні помилки, документи, що використовують мови розмітки можуть не дотримуватись визначеної граматики. Процес перевірки документа на дотримання визначених мовою правил називають **валідацією**, а інструмент, який здійснює перевірку – **валідатором**. Документ, що пройшов цей процес без помилок, називають **валідним**.

Для перегляду HTML-розмітки документа можна використовувати будь-який текстовий редактор. Для перегляду документа, відтвореного за правилами HTML-розмітки, використовується браузер.

HTML документи можуть бути транспортовані так само як і будь-які інші файли (наприклад, за допомогою протоколів FTP, TCP), проте зазвичай вони транспортуються із сервера за допомогою протоколу HTTP або електронною поштою.

Всесвітня павутина складається в основному з HTML-документів, переданих з вебсерверів для браузерів, використовуючи протокол HTTP. До того ж HTTP використовується для передачі зображень, звуків, відео та іншого супутнього контенту. Для правильного відтворення документа браузером окрім нього самого передається ще й інша інформація (метадані), у якій зазвичай міститься визначення MIME типу (наприклад, *text/html* або *application/xhtml+xml*) та кодової таблиці документа.

Ймовірно, HTML – найуспішніша мова розмітки документів у всьому світі.https://uk.wikipedia.org/wiki/HTML Проте, коли світові представили XML, було вирішено створити нову версію HTML, похідну від XML. Адже з XML-заснованим HTML інші XML-мови могли би включати частини XHTML, а XHTML-документи могли б включати частини інших мов розмітки. Також автори вебдокументів могли б скористатися перевагами редизайну задля очищення деяких з найбільш неохайних частин HTML, а також додати деякі з нових необхідних функцій, таких як покращені форми. Нижче зазначені деякі переваги використання XHTML замість HTML.

Якщо документ є лише чистим XHTML 1.0 (не включає інші мови розмітки), то різниця між XHTML та HTML майже не помітна. Проте, оскільки стають доступними все більше і більше XML-інструментів (наприклад, XSLT для перетворення документів), переваги використання XHTML стають все помітнішими. Наприклад, XForms дозволяє досить просто керувати редагуванням документів XHTML (або будь-яких інших видів документа XML). Семантичні вебзастосунки також зможуть скористатися документами XHTML за своїми потребами. Якщо документ містить більш ніж просто XHTML 1.0 (наприклад, у документі використовуються мови розмітки MathML, SMIL, або SVG), тоді переваги використання XHTML значно помітніші, адже HTML не підтримує такі комбінації мов розмітки в одному документі.

R – мова програмування і програмне середовище для статистичних обчислень, аналізу та зображення даних в графічному вигляді. Розробка R відбувалась під істотним впливом двох наявних мов програмування: мови програмування S з семантикою успадкованою від Scheme. R названа за

першою літерою імен її засновників Роса Іхаки (Ross Ihaka) та Роберта Джентлмена (Robert Gentleman) працівників Оклендського Університету в Новій Зеландії.

R поширюється безкоштовно за ліцензією GNU General Public License у вигляді вільнодоступного вихідного коду або відкомпільованих бінарних версій більшості операційних систем: Linux, FreeBSD, Microsoft Windows, Mac OS X, Solaris. R використовує текстовий інтерфейс, однак існують різні графічні інтерфейси користувача (див. Графічні Редактори Скриптів та IDE).

R має значні можливості для здійснення статистичних аналізів, включаючи лінійну і нелінійну регресію, класичні статистичні тести, аналіз часових рядів (серій), кластерний аналіз і багато іншого. R легко розбудовується завдяки використанню додаткових функцій і пакетів доступних на сайті Comprehensive R Archive Network (CRAN). Більша частина стандартних функцій R, написана мовою R, однак існує можливість підключати код написаний C, C++, або Фортраном. Також за допомогою програмного коду на C або Java можна безпосередньо маніпулювати R об'єктами.

R належить до інтерпретованих мов програмування і для роботи використовується командний інтерпретатор.

R підтримує концепцію об'єктно-орієнтованого програмування (ООП) включаючи generic функції, результат виконання якої залежить від аргументів (типу об'єктів), що передаються generic функції. В мові програмування R всі змінні є об'єктами, кожен об'єкт належить до певного класу.

Середовище R містить засоби для візуалізації результатів обчислень (двовимірні, тривимірні графіки, діаграми, гістограми, діаграми (схеми) Ганта тощо). Графічні можливості R дозволяють створювати високоякісні графіки з різними атрибутами, зокрема математичні формули і символи.

R *de-facto* став стандартом у міжнародній спільноті спеціалістів в галузі статистики, і широко використовується в розробках статистичних програм та аналізі даних. Згідно щорічному опитуванню Rexer's Annual Data Miner Survey в 2010 році, більшість (43%) серед опитаних спеціалістів з аналізу даних використовують у своїй роботі середовище R.

Можливості R значно розширюються додатковими пакетами (бібліотеками). Пакети розробляються безпосередньо користувачами R. Існує понад 4500 пакетів, доступних на сайті Comprehensive R Archive Network (CRAN), Omegahat, Bioconductor, R-Forge.

На сторінці "Task View" вебсайту CRAN розміщено список напрямків (Фінанси, Генетика, Хеміометрія і Математична Фізика, Навколишнє середовище, Суспільні науки) в яких використовується R і для яких доступні пакети на сайті.

Для роботи з R існує кілька графічних інтерфейсів (GUI), наприклад:

- Графічна оболонка RGui разом з командною оболонкою (терміналом) R Console входять до базового пакету R у версії для Windows.
- RStudio – зручне кросплатформне середовище розробки з відкритим кодом (існує можливість запуску на віддаленому linux сервері).

- RExcel – додаток до Microsoft Excel, який дозволяє використовувати можливості R.

R доступна для використання у мовах програмуваннях Python (за допомогою пакета RPy, Perl (за допомогою модуля Statistics::R) і Ruby (за допомогою RSRuby).

Деякі пропрієтарні програмні продукти призначені для аналізу статистичних даних (напр. SPSS, STATISTICA[, SAS) мають розширення, розроблені для інтеграції у свої структури функціоналу R.

Контрольні запитання

1. *Визначення мовам програмування.*
2. *Класифікація мов програмування.*
3. *Характеристика мови програмування BASIC.*
4. *Характеристика мови програмування* Python.
5. *Характеристика мови програмування Java.*
6. *Характеристика мови програмування JavaScript.*
7. *Характеристика мови програмування HTML.*
8. *Характеристика мови програмування R.*

Контрольні запитання
(зробити посилання відразу на тести для електронного варіанту методички!!!!

СЕРЕДОВИЩЕ ПРОГНАМУВАННЯ

***Анотація**. Для написання програми на одній із мов програмування використовують редактор початкового коду. Редактор може бути окремим додатком або інтегрованим у середовище розробки. Для швидкого і зручного створювання програм з графічним інтерфейсом використовують спеціальні середовища візуального програмування*: *Microsoft Visual Basic*; *Microsoft Visual Studio*; *Power Apps.*

Для написання комп'ютерних програм на будь-якій мові програмування потрібен редактор для їх набору. **Редактор початкового коду** – текстовий редактор для створення, та редагування початкового коду програм. Він може бути окремим додатком, або інтегрованим в інтегроване середовище розробки (IDE). Редактори програмного коду мають деякі можливості, які спрощують та прискорюють написання та редагування коду, такі як підсвітка синтаксису, автодоповнення, перевірка правильності розстановки дужок, структурний видрук, контекстна допомога по коду та багато інших. Такі редактори надають зручний спосіб для запуску компілятора, інтерпретатора, налагоджувача або інших програм необхідних у процесі розробки програмного забезпечення. Незважаючи на те, що багато текстових редакторів можуть бути використані

для редагування тексту програм, якщо вони не мають розширених можливостей, що автоматизують або спрощують введення та модифікацію коду, то вони не можуть називатися «редакторами початкового коду», а просто є «текстовими редакторами», які також можуть бути використані для редагування програм.

Для швидкого і зручного створювання програм з графічним інтерфейсом використовують спеціальні середовища візуального програмування. **Візуальне середовище програмування** – інтегроване середовище розробки програмних засобів (IDE, яке містить редактор вихідного коду, компілятор чи/або інтерпретатор, засоби автоматизації збірки та засоби для спрощення розробки графічного інтерфейсу користувача). Візуальне програмування ще називають Rapid Application Development (RAD), “швидка розробка додатків”. Технологія RAD суттєво прискорює створення програм з графічним інтерфейсом.

Середовища для візуального програмування також надають змогу конструювати програми шляхом оперування графічними об’єктами. Багато сучасних візуальних середовищ програмування використовуються для реалізації принципів об'єктно-орієнтованого підходу у розробці програмного забезпечення.

Інтегровані середовища розробки створені для того, щоб максимізувати продуктивність програміста, надавши йому пов'язані інструменти розробки зі схожими інтерфейсами як одну програму, в якій відбуватиметься весь процес розробки й яка надає необхідні функції для модифікації, компілювання, розгортання та налагодження програмного забезпечення. Протилежним до цього є підхід до розробки ПЗ, під час якого використовуються окремі інструменти, так як vi, GCC або make.

Одним із завдань ІСР є зменшення часу, необхідного на конфігурацію різноманітних інструментів розробки, натомість пропонуючи той самий набір, як єдине ціле. Це може збільшити продуктивність розробника, у випадку, коли навчання тому, як працює інтегроване середовище розробки є швидшим, ніж освоєння всіх інструментів зокрема. Крім того, більша інтеграція між вбудованими інструментами потенційно може сприяти додатковому збільшенню продуктивності. Наприклад, синтаксичний аналіз коду може відбуватися безпосередньо під час його редагування, тим самим виявляючи помилки ще до трансляції коду.

Деякі інтегровані середовища розробки призначені для використання певної мови програмування (або декількох споріднених мов), надаючи набір можливостей, які більш підходять до парадигми програмування відповідної мови. Такими ІСР є, наприклад PhpStorm, Xcode, Xojo та Delphi.

З іншої сторони, існує чимало більш універсальних ІСР, які є багатомовними, наприклад Eclipse, ActiveState Komodo, IntelliJ IDEA, MyEclipse, Oracle JDeveloper, NetBeans Codenvy and Microsoft Visual Studio.

Переважна більшість нинішніх ІСР мають графічний інтерфейс користувача. До появи систем, які підтримують вікна, таких як Microsoft Windows та X Window System (X11), широко використовувалися консольні

інтегровані середовища розробки, такі як Turbo Pascal. Особливою їх прикметою було широке використання функціональних клавіш та поєднань клавіш для запуску команд або макросів, які часто використовувалися.

Microsoft Visual Basic – засіб розробки програмного забезпечення, створений і підтримуваний корпорацією Microsoft, який складається з мови програмування і середовища розроблення. Мова Visual Basic успадкувала дух, стиль і, частково, синтаксис свого предка – мови Бейсік, яка має чимало діалектів. Водночас Visual Basic поєднує в собі процедури, елементи об'єктно-орієнтованих та компонентно-орієнтованих мов програмування. Середовище розробки VB містить інструменти для візуального конструювання користувацького інтерфейсу.

Visual Basic вважається потужним засобом швидкої розробки прототипів програми, розробки застосунків, що працюють з базами даних і взагалі для компонентного способу створення програм, що працюють під управлінням майже усіх версій операційних систем сімейства Microsoft Windows.

Перше визнання серйозними розробниками Visual Basic отримав після виходу версії 3 – VB3. Остаточне визнання як повноцінного засобу програмування для Windows – при виході версії 5 – VB5. Версія VB6, що входить до складу Microsoft Visual Studio 6.0, стала по-справжньому зрілим і функціонально багатим продуктом. Після цього розробники з Microsoft суттєво змінили напрямок розвитку даної технології.

Visual Basic.NET не дозволяє програмувати по-старому, бо по суті є зовсім іншою мовою, такою ж, як і будь-яка інша мова програмування для платформи .NET. Індивідуальність мови і її переваги (простота, природність створення програм, легкість використання готових компонент) при використанні в середовищі .NET не мають такого значення, як раніше – усе зосереджено на можливостях самої системи .NET, на її бібліотеці класів. Тому нині треба розрізняти класичний Visual Basic з його діалектами (Visual Basic for Applications (VBA) і Visual Basic Scripting Edition (VBScript)) і мову програмування для платформи .NET – Visual Basic.NET.

Класичний Visual Basic (версії 5-6) – ця мова дуже сильно прив'язана до свого середовища розробки й до операційної системи Windows, оскільки вона є виключно інструментом написання Windows-застосунків. Прив'язаність до середовища полягає в тому, що існує велика кількість засобів, призначених для допомоги й зручності програмування: вбудований зневаджувач, перегляд змінних і структур даних на льоту, вікно зневадження, спливна підказка при наборі тексту програми (Intellisense). Усі ці переваги роблять марним і навіть неможливим використання Visual Basic поза середовищем для розроблення, наприклад, у звичайному текстовому редакторі.

Visual Basic for Applications (VBA) – це засіб програмування, який практично нічим не відрізняється від класичного Visual Basic, і призначений для написання макросів та інших прикладних програм для конкретних програм. Найбільшу популярність здобув завдяки своєму використанню в пакеті Microsoft Office. З самого початку широке розповсюдження Visual Basic

for Applications, у поєднанні з недостатньою увагою до питань безпеки, призвело до значного поширення макровірусів.

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio включає один або декілька з наступних компонентів:

- Visual Basic .NET, а до його появи – Visual Basic;
- Visual C++;
- Visual C#;
- Visual F# (входить до складу Visual Studio 2010);
- Visual Studio Debugger.

Багато варіантів постачання також містять:

- Microsoft SQL Server або
- MSDE Visual Source Safe – файл-серверна система керування версіями.

Visual Studio 2022 – інтегроване середовище розробки Visual Studio є творчим стартовим майданчиком, який можна використовувати для редагування, налагодження та складання коду, а також для публікації програми. На додаток до стандартного редактора та відладчика, що надаються більшістю інтегрованих середовищ розробки, Visual Studio 2022 включає компілятори, засоби завершення коду, графічні конструктори та багато інших функцій для покращення процесу розробки програмного забезпечення.

Visual Studio Community – безкоштовне повнофункціональне середовище IDE, що розширюється, для створення сучасних програм Android, iOS і Windows, а також веб-додатків і хмарних служб.

Visual Studio Professional – це потужне повнофункціональне інтегроване середовище розробки для програмістів, які створюють та розгортають інноваційні програми для будь-якої платформи.

Visual Studio Enterprise – це потужна, повнофункціональна IDE для розробників, які розробляють, тестують та розгортають складні програми для будь-якої платформи, включаючи платформи Майкрософт.

Visual Studio побудована в архітектурі, що підтримує можливість використання доповнень (Add-Ins), – плагінів від сторонніх розробників, що дає змогу розширювати можливості середовища розробки.

Деякі з найпопулярніших доповнень:

- DevPartner Studio;
- Visual Assist;
- ReSharper;
- IncrediBuild;
- Workspace Whiz;

- Viva64.

Power Apps – це набір програм, послуг і з'єднувачів, а також платформа даних, яка дозволяє швидко створювати користувацькі програми для бізнес-потреб. За допомогою Power Apps можна швидко створити користувацькі бізнес-програми, які підключаються до даних, що зберігаються в базовій платформі даних (Microsoft Dataverse) або в різних онлайнових і локальних джерелах даних, таких як SharePoint, Microsoft 365, Dynamics 365, SQL Server тощо.

Програми, створені за допомогою Power Apps, дозволяють використовувати широкі можливості бізнес-логіки для перетворення ручних бізнес-операцій на цифрові автоматизовані процеси. Крім того, програми, створені за допомогою Power Apps, мають адаптивний дизайн і можуть без проблем працювати у браузері та на мобільних пристроях (телефоні або планшеті). Power Apps «демократизує» процес створення бізнес-програм, тому для створення багатофункціональних настроюваних бізнес-програм користувачам не потрібно писати код.

Power Apps також надає розширювану платформу, яка дозволяє професійним розробникам програмно взаємодіяти з даними та метаданими, застосовувати бізнес-логіку, створювати настроювані з'єднувачі та інтегруватись із зовнішніми даними.

RStudio – вільне та відкрите інтегроване середовище розробки (IDE) для R, мови програмування обчислювальної статистики та візуалізації даних. RStudio була започаткована Джозефом Аллером, творцем мови програмування ColdFusion . Хадлі Вікхем головний науковець RStudio.

RStudio доступна в двох версіях: *RStudio Desktop*, в якій програми працюють як звичайні застосунки та *RStudio Server*, яка дозволяє отримувати доступ до RStudio, запущеної на віддаленому Linux сервері, через вебінтерфейс. Дистрибутиви RStudio Desktop доступні для Windows, OS X, та Linux.

RStudio доступна у вільній та комерційній редакціях та працює на настільних комп'ютерах (Windows, OS X, та Linux) або у браузері з'єднаному з *RStudio Server* або з *RStudio Server Pro* (Debian, Ubuntu, Red Hat Linux, CentOS, openSUSE та SLES).

RStudio написана на мові C++ з використанням Qt framework для графічного інтерфейсу користувача.[

Розробка RStudio почалась приблизно у грудні 2010, і перша публічна бета-версія (v0.92) була офіційно анонсована у лютому 2011. Версію 1.0 було випущено 1 листопада 2016.

Контрольні запитання

1. *Що таке редактор початкового коду.*
2. *Що таке інтегроване середовище розробки.*
3. *Що таке візуальне середовище програмування.*
4. *Охарактеризуйте Microsoft Visual Basic.*
5. *Охарактеризуйте Microsoft Visual Studio.*

6. *Охарактеризуйте Power Apps.*
7. *Охарактеризуйте RStudio.*

АВТОМАТИЗАЦІЯ ОБЧИСЛЕНЬ В MS EXCEL ЗАДОПОМОГОЮ МАКРОСІВ

Анотація. *Макрос – програмний алгоритм дій записаний користувачем. Призначення макрокоманд полягає в автоматизації часто вживаних послідовностей дій чи команд. Макропрограмами часто називають спеціальні програми, які призначено для виконання всередині інших програм. Наприклад, текстовий редактор Microsoft Word дозволяє виконувати у своєму середовищі програми, що написані спеціальною мовою Visual Basic for Applications (VBA). Макрос можна записати двома способами: автоматично та вручну.*

Макрокоманда, **макро** або **макрос** (множина від англ. *macro*) – програмний алгоритм дій, записаний користувачем.

У програмуванні макрокомандою називають таку абстракцію, коли усі випадки появи у документі тексту, що підпадає під заданий шаблон, модифікуються за заданими правилами. Інтерпретатор або компілятор автоматично змінює такі частини тексту коли на них натрапляє. У мовах, що компілюються така заміна завжди відбувається під час компіляції. Також назва «макро» може вживатися у багатьох інших контекстах, як то клавіатурні макрокоманди та мови макрокоманд тощо. Більшість із таких контекстів безпосередньо пов'язана з тією самою концепцією – одна подана коротка команда або дія при виконанні розгортається у велику кількість інструкцій нижчого рівня абстракції.

Призначення макрокоманд полягає або в автоматизації часто вживаних послідовностей дій чи команд, або у сильнішому абстрагуванні дій/команд.

Наприклад, у Вікіпедії макрокоманда {{subst:PAGENAME}} розкривається інтерпретатором на цій сторінці у «Макрокоманда».

Також макро або макропрограмами часто називають спеціальні програми, які призначено для виконання всередині інших програм. Наприклад, текстовий редактор Microsoft Word дозволяє виконувати у своєму середовищі програми, що написані спеціальною мовою VBA (Visual Basic for Applications). Такі програми звичайно використовуються для виконання типових для макрокоманд задач: автоматизації часто вживаних або складних послідовностей дій користувача.

Порівняно висока ефективність та розвинута функціональність VBA поєднані з можливістю за певних умов автоматичного (без прямої вказівки на те користувачем) виконання макрокоманд призвели до появи та поширення макровірусів.

Кожна з прикладнихпрограм Microsoft Office містить велику кількість об'єктів, що відрізняються один від одного. Кожний з об'єктів має свої властивості, методийподії. Хоча основні властивості VBA залишаються незмінними для різних прикладних програм Microsoft Office (вони об'єднані базовими операторами мови VB), але при цьому кожна прикладна програма

вносить свої специфічні команді об'єкти. Це й відрізняє VBA від універсального середовища програмування VB. Крімтого, програми, розроблені в середовищі VB, після компіляції можуть експлуатуватися не залежно від середовища розробки, а програми, розробленів VBA, є компонентом певного документа однієї з прикладних програм, і вони можуть виконуватися тільки в середовищі своєї прикладної програми. Таким чином, програма, розробленана VBA, входить до складу деякого документу Microsoft Office і слугує для автоматизації оброблення і нформації в рамках цього документа.

Зазвичай в прикладній програмі (Microsoft Word, Excel, PowerPoint, Access) є вбудований редактор VB. Удеяких програмах (OutLook, Internet Explorer) використається скороченаверсія VBA – VBA Script.

Вивченнямови VBA буде направлено стосовно прикладної програми Microsoft Office – Microsoft Excel.

Інструментальне середовище Microsoft Excel дозволяє написати макроси, які дозволяють прискорити рутинну роботу, настроїти прикладну програму таким чином, щоб всі інструменти були під рукою, і тим самим підвищити ефективність роботи за рахунок того, що багато операцій, які доводилося робити вручну, будуть виконуватися автоматично.

Макрос можна записати двома способами:

- автоматично;
- вручну.

Скориставшись першим варіантом, ви просто записуєте певні дії в програмі Microsoft Excel, які виконуєте в даний момент часу. Потім, можна буде відтворити цей запис. Даний спосіб дуже легкий, і не вимагає знання коду, але застосування його на практиці досить обмежене.

Ручна зйомка макросів, навпаки, вимагає знань програмування, так як код набирається вручну з клавіатури. Але, грамотне написання таким чином коду, може значно прискорити виконання процесів.

Перш, ніж почати автоматичну запис макросів, потрібно включити макроси в програмі Microsoft Excel.

Далі, переходимо у вкладку «Розробник». Кількома по кнопці «Запис макросу», яка розташована на стрічці в блоці інструментів «Код».

Відкривається вікно налаштування запису макросу. Тут можна вказати будь-яке ім'я макросу, якщо встановлене за замовчуванням вас не влаштовує. Головне, щоб ім'я це починалося з букви, а не з цифри. Також, в назві не повинно бути пробілів. За замовчуванням назва буде – «Макрос1», «Макрос2» і т.д...

Тут же, при бажанні, можна встановити поєднання клавіш, при натисканні на які макрос буде запускатися. Першою клавішею обов'язково повинна бути клавіша Ctrl, а другу клавішу користувач встановлює самостійно. Наприклад, клавішу M.

Далі, потрібно визначити, де буде зберігатися макрос. За замовчуванням, він буде зберігатися в цій же книзі, але при бажанні можна встановити зберігання в новій книзі, або в окремій книзі макросів.

У самому нижньому полі настройки макросів можна залишити будь-який придатний по контексту опис даного макросу. Але, це робити не обов'язково.

Коли всі налаштування виконані, можна натиснути на кнопку «ОК».

Після цього, всі ваші дії в даній книзі Excel будуть записуватися в макрос до тих пір, поки ви не зупините запис.

Для прикладу, запишіть найпростіше арифметичне дію: додавання вмісту трьох осередків (= C4 + C5 + C6).

Після цього, тиснемо на кнопку «Зупинити запис». Ця кнопка перетворилася з кнопки «Запис макросу», після включення запису.

Для того, щоб перевірити, як працює записаний макрос, натискаємо в тому ж блоці інструментів «Код» по кнопці «Макрос», або тиснемо клавіші Alt + F8.

Після цього, відкривається вікно зі списком записаних макросів. Шукаємо макрос, який ми записали, виділяємо його, і тиснемо на кнопку «Виконати».

Можна вчинити ще простіше, і не викликати навіть вікно вибору макросів. Ми ж пам'ятаємо, що записали поєднання «гарячих клавіш» для швидкого виклику макросу. У нашому випадку, це Ctrl + M. Набираємо цю комбінацію на клавіатурі, після чого макрос запускається.

Як бачимо, макрос виконав в точності всі ті дії, які були записані раніше.

Для того, щоб відредагувати макрос, знову тиснемо на кнопку «Макрос». У вікні вибираємо потрібний макрос, і натискаємо на кнопку «Змінити».

Відкривається Microsoft Visual Basic (VBE) – середовище, де відбувається редагування макросів.

Запис кожного макросу починається з команди Sub, а закінчується командою End Sub. Відразу ж після команди Sub вказується ім'я макросу. Оператор «Range (« ... »). Select» вказує вибір осередку. Наприклад, при команді «Range (« C4 »). Select» вибирається осередок C4. Оператор «ActiveCell.FormulaR1C1» використовується для запису дій в формулах, і для інших розрахунків.

Спробуємо трохи змінити макрос. Для цього, в макрос допишемо вираз:Range («C3»). Select ActiveCell.FormulaR1C1 = «11»

Вираз «ActiveCell.FormulaR1C1 =« = R [-3] C + R [-2] C + R [-1] C »» замінимо на «ActiveCell.FormulaR1C1 =« = R [-4] C + R [-3] C + R [-2] C + R [-1] C »».

Закриваємо редактор, і запускаємо макрос, як і в минулий раз. Як бачимо, внаслідок введених нами змін був доданий ще один осередок з даними. Він також був включена в розрахунок загальної суми.

У разі, якщо макрос занадто великий, його виконання може зайняти значний час. Але, шляхом внесення ручної зміни в код, ми можемо прискорити процес. Додаємо команду «Application.ScreenUpdating = False». Вона дозволить зберегти обчислювальні потужності, а значить прискорити роботу.

Це досягається шляхом відмови від оновлення екрану під час виконання обчислювальних дій. Щоб відновити оновлення після виконання макросу, в його кінці пишемо команду «Application.ScreenUpdating = True».

Додамо також команду «Application.Calculation = xlCalculationManual» спочатку коду, а в кінці коду дописуємо «Application.Calculation =

xlCalculationAutomatic». Цим ми спочатку макросу відключаємо автоматичний перерахунок результату після кожної зміни осередків, а в кінці макросу – включаємо. Таким чином, Excel підрахує результат тільки один раз, а не буде його постійно перераховувати, чим заощадить час.

Просунуті користувачі можуть виконувати не тільки редагування і оптимізацію записаних макросів, а й записувати код макросів з нуля. Для того, щоб приступити до цього, потрібно натиснути на кнопку «Visual Basic», яка розташована в самому початку стрічки розробника.

Після цього, відкривається знайоме нам вікно редактора VBE. Програміст пише там код макросу вручну.

Як бачимо, макроси в Microsoft Excel можуть значно прискорити виконання рутинних і одноманітних процесів. Але, в більшості випадків, для цього більше підходять макроси, код яких написаний вручну, а не автоматично записані дії. Крім того, код макросу можна оптимізувати через редактор VBE для прискорення процесу виконання завдання.

Контрольні запитання

1. *Що таке макрос.*
2. *Чим відрізняється VBA від універсального середовища програмування VB.*
3. *Чим відрізняється автоматизований спосіб записування макросу від ручного.*
4. *Особливості запису макросів у Microsoft Excel.*
5. *Охарактеризуйте редактор VBE. (Перевірити чи вірно буква Е чи треба А)*

Практичне завдання

1. *Створіть приклад автоматичного запису макросу в Microsoft Excel.*
2. *За допомогою ручного запису макросу зробіть проект калькулятор у Microsoft Excel.*

ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ VISUAL STUDIO

(https://visualstudio.microsoft.com/ru/vs/getting-started/)

Анотація. *Microsoft Visual Studio – інтегроване середовище розробки програмного забезпечення. Дає змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки та вебслужби. Visual Studio 2022 існує в трьох версіях. Visual Studio Community – безкоштовне повнофункціональне середовище IDE. Visual Studio Professional – для програмістів, які створюють та розгортають інноваційні програми для будь-якої платформи. Visual Studio Enterprise – для розробників, які розробляють, тестують та розгортають складні програми для будь-якої платформи, включаючи платформи Майкрософт.*

Visual Studio Community надається безкоштовно для вивчення та індивідуального використання. Спочатку скачайте та встановіть останню версію Visual Studio. Ви можете заощадити час встановлення та місце на диску, обравши лише необхідні компоненти. За потреби ви можете поетапно додавати додаткові компоненти у будь-який час.

За допомогою Visual Studio та .NET можна розробляти класичні програми, веб-програми, мобільні програми, ігри та рішення для Інтернету речей. Програми .NET можна створювати мовою C#, F# або Visual Basic.

Розробка класичних додатків за допомогою .NET Core

Консольна програма – це комп'ютерна програма, розроблена для використання лише через текстовий інтерфейс комп'ютера, такий як текстовий термінал, інтерфейс командного рядка деяких операційних систем (Unix, DOS, тощо) або текстовий інтерфейс, що входить до більшості операційних систем із графічним інтерфейсом користувача (GUI), таких як консоль Windows у Microsoft Windows, термінал у macOS та xterm у Unix.

Зазвичай користувач взаємодіє з консольною програмою, використовуючи лише клавіатуру та екран дисплея, на відміну від програм із графічним інтерфейсом користувача, для яких зазвичай потрібна миша чи інший вказівний пристрій. У міру того, як швидкість і простота використання графічних інтерфейсів програм з часом покращилися, використання консольних програм значно зменшилося, але не зникло.

Створення проекту консольної програми .NET під назвою «HelloWorld».

1. Запустіть Visual Studio 2022.
2. На початковій сторінці виберіть **Створити новий проект**.
3. На сторінці **Створення нового проекту** введіть **консоль** у полі пошуку. Далі виберіть **C#** або **Visual Basic** зі списку мов, а потім виберіть **Усі платформи** зі списку платформ. Виберіть шаблон **консольної програми**, а потім виберіть **Далі**.
4. У діалоговому вікні «**Налаштувати новий проект**» введіть «**HelloWorld**» у полі **«Назва проекту»**. Потім виберіть **Далі** .
5. У діалоговому вікні **Додаткова інформація**:

- Виберіть **.NET 7 (стандартна підтримка)**.
- Виберіть **Не використовувати заяви верхнього рівня**.
- Виберіть **Створити** .

Шаблон створює просту програму, яка відображає «Hello World» у вікні консолі. Код міститься у файлі *Program.cs* або *Program.vb:*

```
Imports System
Module Program
    Sub Main(args As String())
        Console.WriteLine("Hello World!")
    End Sub
End Module
```

Якщо мова, якою ви хочете користуватися, не відображається, змініть селектор мови у верхній частині сторінки.

Код визначає клас Program з одним методом, Main який приймає масив String як аргумент. Main це точка входу програми, метод, який автоматично викликається середовищем виконання під час запуску програми. Будь-які аргументи командного рядка, які надаються під час запуску програми, доступні в масиві *args.*

В останній версії C# нова функція під назвою оператори верхнього рівня дозволяє опускати Program клас і Main метод. Більшість існуючих програм на C# не використовують оператори верхнього рівня, тому в цьому прикладі ця нова функція не використовується. Але він доступний у C# 10, і чи використовуватимете ви його у своїх програмах, залежить від стилю.

Запуск програми:

1. Натисніть Ctrl+ F5, щоб запустити програму без налагодження.

Відкриється вікно консолі з текстом "Hello World!" надруковані на екрані.

2. Натисніть будь-яку клавішу, щоб закрити вікно консолі.

Удосконалити програму, щоб запитувати у користувача своє ім'я та відображати його разом із датою та часом.

1. У *Program.cs* або *Program.vb* замініть вміст Main методу, який є рядком, який викликає Console.WriteLine, таким кодом:

```
Console.WriteLine("What is your name?")
Dim name = Console.ReadLine()
Dim currentDate = DateTime.Now
Console.WriteLine($"{Environment.NewLine}Hello, {name}, on {currentDate:d} at
{currentDate:t}")
Console.Write($"{Environment.NewLine}Press any key to exit...")
Console.ReadKey(True)
```

Цей код відображає підказку у вікні консолі та чекає, доки користувач введе рядок, а потім Enter ключ. Він зберігає цей рядок у змінній з іменем name. Він також отримує значення властивості DateTime.Now , яке містить поточний місцевий час, і призначає його змінній з іменем currentDate. І він відображає ці значення у вікні консолі. Нарешті, він відображає підказку у вікні консолі та викликає метод Console.ReadKey(Boolean) для очікування введення користувача.

Environment.NewLine – це незалежний від платформи та мови спосіб представлення розриву рядка. Альтернативи є \nв C# і vbCrLfVisual Basic.

Знак долара ($) перед рядком дозволяє розміщувати в рядку такі вирази, як імена змінних у фігурних дужках. Значення виразу вставляється в рядок замість виразу. Цей синтаксис називається інтерпольованими рядками.

2. Натисніть Ctrl+ F5, щоб запустити програму без налагодження.

3. У відповідь на підказку введіть ім'я та натисніть Enter клавішу.

4. Натисніть будь-яку клавішу, щоб закрити вікно консолі.

Розробка веб-додатків та хмарних рішень за допомогою ASP.NET Core

Створення веб-програми Razor Pages

Розробка ігор за допомогою Unity

Машинне навчання за допомогою ML.NET Це складно – поки пропускаємо

Створення консольної програми калькулятор на C++

Установіть і запустіть на вашому комп'ютері Visual Studio з **розробкою робочого столу з C++.** Якщо його ще не встановлено, перегляньте статтю Інсталяція підтримки C++ у Visual Studio.

Якщо ви щойно запустили Visual Studio, ви побачите діалогове вікно Visual Studio Start. Щоб розпочати, виберіть **Створити новий проект.**

1. У іншому випадку на панелі меню у Visual Studio виберіть «**Файл**» > «**Створити**» > «**Проект**». Відкриється вікно **Створити новий проект**.

2. У списку шаблонів проектів оберіть **Консольний додаток**, а потім виберіть **Далі**.

3. У діалоговому вікні «**Налаштувати новий проект**» виберіть поле редагування **«Назва проекту», назвіть свій новий проект** «*CalculatorTutorial*» , а потім виберіть «**Створити**».

Буде створено пусту консольну програму C++ Windows. Консольні програми використовують вікно консолі Windows для відображення виводу та приймання введених даних користувачами. У Visual Studio відкривається вікно редактора, у якому показано згенерований код:

```
// CalculatorTutorial.cpp : This file contains the 'main' function. Program execution begins and ends there.
//
#include <iostream>
int main()
{
    std::cout << "Hello World!\n";
}
// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
// Debug program: F5 or Debug > Start Debugging menu
// Tips for Getting Started:
//   1. Use the Solution Explorer window to add/manage files
//   2. Use the Team Explorer window to connect to source control
//   3. Use the Output window to see build output and other messages
//   4. Use the Error List window to view errors
//   5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project
//   6. In the future, to open this project again, go to File > Open > Project and select the .sln file
```

Шаблон для нової консольної програми Windows створює просту програму «Hello World» C++. На цьому етапі ви можете побачити, як Visual Studio створює та запускає програми, які ви створюєте прямо з IDE.

1. Щоб побудувати свій проект, виберіть **Build Solution** у меню **Build**. У вікні **виведення** відображаються результати процесу збирання.

2. Щоб запустити код, на панелі меню виберіть **Налагодити**, **Почати без налагодження**.

Відкриється вікно консолі, а потім запуститься ваша програма. Коли ви запускаєте консольну програму в Visual Studio, вона виконує ваш код, а потім друкує «Натисніть будь-яку клавішу, щоб закрити це вікно...» щоб дати вам можливість побачити результат. Щиро вітаю! Ви створили свій перший "Hello, world!" консольний додаток у Visual Studio!

3. Натисніть клавішу, щоб закрити вікно консолі та повернутися до Visual Studio.

Тепер у вас є інструменти для створення та запуску програми після кожної зміни, щоб переконатися, що код все ще працює так, як ви очікуєте. Пізніше розглянемо, як налагодити його, якщо це не так.

Тепер давайте перетворимо код у цьому шаблоні на додаток-калькулятор.

1. У *CalculatorTutorial.cpp* файлі відредагуйте код відповідно до цього прикладу:

```
// CalculatorTutorial.cpp : This file contains the 'main' function. Program execution begins and ends there.
//
#include <iostream>
using namespace std;
int main()
{
   cout << "Calculator Console Application" << endl << endl;
   cout << "Please enter the operation to perform. Format: a+b | a-b | a*b | a/b"
      << endl;
   return 0;
}
// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
// Debug program: F5 or Debug > Start Debugging menu
// Tips for Getting Started:
//   1. Use the Solution Explorer window to add/manage files
//   2. Use the Team Explorer window to connect to source control
//   3. Use the Output window to see build output and other messages
//   4. Use the Error List window to view errors
//   5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project
//   6. In the future, to open this project again, go to File > Open > Project and select the .sln file
```

1. Розуміння коду:
   - Інструкції *#include* дозволяють посилатися на код, розташований в інших файлах. Іноді ви можете побачити назву файлу в кутових дужках (<>); в інших випадках він оточений лапками (" "). Загалом кутові дужки використовуються під час посилань на стандартну бібліотеку C++, тоді як лапки використовуються для інших файлів.
   - Рядок *using namespace std;* вказує компілятору очікувати використання в цьому файлі матеріалу зі стандартної бібліотеки C++. Без цього рядка кожному ключовому слову з бібліотеки мав би передувати *std::*, щоб позначити його область. Наприклад, без цього рядка кожне посилання на *cout* потрібно було б записати як *std::cout.* Інструкція **using** додається, щоб зробити код більш зрозумілим.
   - Ключове *cout* слово використовується для друку на стандартний вивід у C++. Оператор << повідомляє компілятору відправити все, що знаходиться праворуч від нього, на стандартний вивід.
   - Ключове слово **endl** схоже на клавішу *Enter*; він завершує рядок і переміщує курсор на наступний рядок. Краще вставити *\n* всередину рядка (міститься в «»), щоб зробити те саме, оскільки *endl* завжди очищає буфер і може зашкодити продуктивності програми, але оскільки це дуже маленька програма, *endl* використовується натомість для кращої читабельності.
   - Усі оператори C++ мають закінчуватися крапкою з комою, а всі програми C++ мають містити *main()* функцію. Ця функція – це те, що

програма запускає на початку. Весь код має бути доступним *main()* для того, щоб його можна було використовувати.

2. Щоб зберегти файл, **натисніть Ctrl+S** або виберіть піктограму «**Зберегти** » у верхній частині IDE, піктограму дискети на панелі інструментів під рядком меню.

3. Щоб запустити програму, натисніть **Ctrl+F5** або перейдіть до меню **Debug** і виберіть **Start Without Debugging**. Ви повинні побачити вікно консолі, яке відображає текст, указаний у коді.

4. Коли закінчите, закрийте вікно консолі.

Настав час додати трохи математичної логіки. Щоб додати клас Calculator:

1. Перейдіть до меню «**Проект**» і виберіть «**Додати клас**». У полі редагування «**Назва класу**» введіть «*Калькулятор*». Виберіть **OK**. До вашого проекту буде додано два нових файли. Щоб зберегти всі змінені файли одночасно, натисніть **Ctrl+Shift+S**. Це комбінація клавіш для «**Файл**» > «**Зберегти все**». На панелі інструментів також є кнопка «**Зберегти все**» – піктограма двох дискет поруч із кнопкою «**Зберегти**». Загалом, добре виконувати функцію «**Зберегти все**» часто, щоб не пропустити файли під час збереження.

Клас схожий на план для об’єкта, який щось робить. У цьому випадку ми визначаємо калькулятор і як він має працювати. Майстер **додавання класу**, який ви використали вище, створив файли *.h і .cpp*, які мають таке ж ім’я, що й клас. Ви можете побачити повний список файлів проекту у вікні **Solution Explorer**, видимому збоку IDE. Якщо вікно не відображається, ви можете відкрити його з панелі меню: виберіть «**Перегляд**» > «**Провідник рішень**».

Тепер у редакторі має бути відкрито три вкладки: *CalculatorTutorial.cpp*, *Calculator.h*і *Calculator.cpp*. Якщо ви випадково закрили один із них, ви можете знову відкрити його, двічі клацнувши його у вікні **Solution Explorer**.

2. У ***Calculator.h***, видаліть рядки *Calculator()*;і *~Calculator();*, які були згенеровані, оскільки вони вам тут не знадобляться. Далі додайте наступний рядок коду, щоб файл виглядав так:

```
#pragma once
class Calculator
{
public:
   double Calculate(double x, char oper, double y);
};
```

Розуміння коду:

- Рядок, який ви додали, оголошує нову функцію під назвою *Calculate*, яку ми будемо використовувати для виконання математичних операцій додавання, віднімання, множення та ділення.
- Код C++ організований у файли *заголовків* (*.h*) і *вихідні* файли (*.cpp*). Кілька інших розширень файлів підтримуються різними компіляторами, але це основні, про які слід знати. Функції та змінні зазвичай *оголошуються*, тобто їм надається ім’я та тип, у файлах заголовків, і

реалізуються, або надається визначення, у вихідних файлах. Щоб отримати доступ до коду, визначеного в іншому файлі, ви можете використовувати #include "*filename.h*", де '*filename.h*' – це ім'я файлу, який оголошує змінні або функції, які ви хочете використовувати.

- Два рядки, які ви видалили, оголосили *конструктор* і *деструктор* для класу. Для такого простого класу, як цей, компілятор створює їх для вас, і їх використання виходить за рамки цього підручника.
- Рекомендується організовувати код у різні файли залежно від того, що він робить, щоб потім було легко знайти потрібний код. У нашому випадку ми визначаємо *Calculator* клас окремо від файлу, що містить *main()* функцію, але ми плануємо посилатися на *Calculator* клас у *main()*.

3. Ви побачите, що під *Calculate*. Це тому, що ми не визначили *Calculate* функцію у файлі *.cpp*. Наведіть вказівник миші на слово, клацніть лампочку (у цьому випадку викрутку), яка з'явиться, і виберіть **Створити визначення «Обчислити» в Calculator.cpp**.

З'являється спливаюче вікно, яке дає змогу переглянути зміну коду, внесену в інший файл. Код було додано до *Calculator.cpp*.

Наразі він повертає лише 0,0. Давайте це змінимо. Натисніть **Esc** , щоб закрити спливаюче вікно.

4. Перейдіть до *Calculator.cpp* файлу у вікні редактора. Видаліть розділи *Calculator()* та *~Calculator()* (як у файлі *.h*) і додайте наступний код до *Calculate():*

```
#include "Calculator.h"
double Calculator::Calculate(double x, char oper, double y)
{
   switch(oper)
   {
      case '+':
         return x + y;
      case '-':
         return x - y;
      case '*':
         return x * y;
      case '/':
         return x / y;
      default:
         return 0.0;
   }
}
```

Розуміння коду:

- Функція Calculate використовує число, оператор і друге число, а потім виконує запитану операцію над числами.
- Оператор *switch* перевіряє, який оператор було надано, і виконує лише регістр, що відповідає цій операції. Значення за замовчуванням: *case* є запасним варіантом у випадку, якщо користувач вводить оператор, який не приймається, тому програма не порушує роботу. Загалом,

найкраще обробляти недійсний введений користувачем елегантніший спосіб, але це виходить за рамки цього посібника.

- Ключове **double** слово позначає тип числа, який підтримує десяткові знаки. Таким чином, калькулятор може обробляти як десяткову математику, так і цілу математику. Функція *Calculate* повинна завжди повертати таке число через те, що **double** на самому початку коду (це позначає тип повернення функції), тому ми повертаємо 0,0 навіть у випадку за замовчуванням.
- Файл *.h* оголошує *прототип* функції, який заздалегідь повідомляє компілятору, які параметри йому потрібні та який тип результату очікувати від нього. Файл *.cpp* містить усі деталі реалізації функції.

Якщо ви створите та запустите код знову на цьому етапі, він усе одно завершить роботу після запиту, яку операцію виконати. Далі ви змініте *main* функцію, щоб виконати деякі обчислення.

1. Тепер оновимо *main* функцію в *CalculatorTutorial.cpp*:

```
// CalculatorTutorial.cpp : This file contains the 'main' function. Program execution begins and ends there.
//
#include <iostream>
#include "Calculator.h"
using namespace std;
int main()
{
   double x = 0.0;
   double y = 0.0;
   double result = 0.0;
   char oper = '+';
   cout << "Calculator Console Application" << endl << endl;
   cout << "Please enter the operation to perform. Format: a+b | a-b | a*b | a/b"
       << endl;
   Calculator c;
   while (true)
   {
      cin >> x >> oper >> y;
      result = c.Calculate(x, oper, y);
      cout << "Result is: " << result << endl;
   }
   return 0;
}
```

Розуміння коду:

- Оскільки програми C++ завжди починаються з *main()* функції, нам потрібно викликати інший код звідти, тому потрібен *#include* оператор.
- Деякі початкові змінні *x*, *y*, *oper* та *result* оголошено для зберігання першого числа, другого числа, оператора та кінцевого результату відповідно. Завжди доцільно надавати їм деякі початкові значення, щоб уникнути невизначеної поведінки, що й робиться тут.

- Рядок *Calculator c*; оголошує об'єкт з іменем *'c'* як екземпляр Calculator класу. Сам клас – це лише схема того, як працюють калькулятори; об’єкт – це конкретний калькулятор, який виконує математику.
- Інструкція *while* (*true*) є циклом. Код усередині циклу продовжує виконуватися знову і знову, доки () виконується умова всередині. Оскільки умова просто вказана як **true**, вона завжди вірна, тому цикл працює вічно. Щоб закрити програму, користувач повинен вручну закрити вікно консолі. В іншому випадку програма завжди чекає нового введення.
- Ключове *cin* слово використовується для прийняття введених даних від користувача. Цей вхідний потік достатньо розумний, щоб обробити рядок тексту, введений у вікні консолі, і розмістити його в кожній із перелічених змінних у порядку, припускаючи, що введені користувачем дані відповідають необхідній специфікації. Ви можете змінити цей рядок, щоб він приймав різні типи введення, наприклад, більше двох чисел, хоча *Calculate()* функцію також потрібно оновити, щоб впоратися з цим.
- Вираз *c.Calculate (x, oper, y)*; викликає *Calculate* функцію, визначену раніше, і надає введені вхідні значення. Потім функція повертає число, яке зберігається в *result*.
- Нарешті, *result* друкується на консолі, щоб користувач бачив результат обчислення.

Тепер настав час знову протестувати програму, щоб переконатися, що все працює належним чином.

1. Натисніть **Ctrl+F5**, щоб перебудувати та запустити програму.
2. Enter 5 + 5 і натисніть **Enter**. Переконайтеся, що результат 10.

Створення програми WinForms на Visual Basic

Windows Forms – інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft.NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді.

Створіть проект Visual Basic. Для цього типу проекту вже є всі необхідні файли шаблонів, що позбавляє вас зайвої роботи.

1. Запустіть Visual Studio.
2. На початковому екрані виберіть **Створити проект**.
3. У вікні **Створити проект** виберіть шаблон **Windows Forms (.NET Framework)** для Visual Basic.

Ви можете уточнити умови пошуку, щоб швидко перейти до потрібного шаблону. Наприклад, введіть *Windows Forms* у полі пошуку. Потім виберіть **Visual Basic** у **списку мов** і **Windows** у **списку платформ**.

4. У полі **Ім'я проекту** вікна **Налаштувати новий проект** введіть *HelloWorld* . Потім виберіть **Створити**.

Коли ви виберете шаблон проекту Visual Basic і задайте ім'я файлу, Visual Studio відкриває форму. Форма є інтерфейсом Windows. Ви створите програму Hello World, додавши до форми елементи керування.

1. У лівій частині інтегрованого середовища розробки Visual Studio виберіть вкладку **Панель елементів**. Якщо її не бачите, виберіть пункт **Представлення** > **Панель елементів** у рядку меню або скористайтеся комбінацією клавіш **CTRL** + **ALT** + **X** .

Якщо потрібно, виберіть піктограму **Закріпити** , щоб закріпити вікно **Панель елементів** .

2. Виберіть елемент керування **Кнопка** та перетягніть його на форму.

3. У розділі **Зовнішній вигляд** вікна **Властивості** введіть для властивості **Text** *Натиснути це* та натисніть клавішу **Enter** .

Якщо вікно **Властивості** не відображається, його можна відкрити у рядку меню. Для цього виберіть **Вигляд** > **Вікно властивостей** або натисніть клавішу **F4** .

4. У розділі **Проектування** вікна **Властивості** змініть ім'я з **Button1** на *btnClickThis* , а потім натисніть клавішу **Enter** .

Тепер, коли ми додали елемент керування "Кнопка" для створення дії, додайте елемент керування "Мітка", куди можна надсилати текст.

5. Виберіть елемент керування **Мітка** у вікні **Панель елементів** , а потім перетягніть його на форму. Розташуйте його під кнопкою **Натиснути це** .

6. У розділі **Проект** або **(DataBindings)** вікна **Властивості** змініть ім'я **Label1** на *lblHelloWorld* та натисніть клавішу **Enter** .

7. У вікні **Form1.vb [Конструктор]** двічі **клацніть цю** кнопку, щоб відкрити вікно **Form1.vb** .

Крім того, можна розгорнути вузол **Form1.vb** в браузері **рішень** , а потім вибрати **Form1** .

8. У вікні **Form1.vb** між **приватними вкладеними та кінцевими вкладеними** рядками введіть *lblHelloWorld.Text = "Hello World!"*, як показано на наступному знімку екрана:

Програма готова до складання та запуску.

1. Виберіть **Пуск** , щоб запустити програму.

У разі відбувається таке. В інтегрованому середовищі розробки Visual Studio відкриваються вікна **Засоби діагностики** та **Висновок**. Поза цим середовищем відкриється діалогове вікно **Form1**. Воно буде містити кнопку **Натиснути це** і текст **Label1** .

2. Натисніть кнопку **Натисніть це** у діалоговому вікні **Form1** .

Текст **Label1** змінюється на **Hello World!** .

3. Закрийте діалогове вікно **Form1** для завершення роботи програми.

Контрольні запитання

1. *Дайте визначення, що таке консольна програма.*
2. *Опишіть процес створення консольної програми.*
3. Дайте визначення, що таке *Windows Forms.*
4. *Опишіть процес створення програми WinForms.*

Практичне завдання

1. *Створіть приклад класичного додатку за допомогою .NET Core.*
2. *Створіть програму WinForms на Visual Basic.*

СТВОРЕННЯ ТА ПРОГРАМУВАННЯ WEB-СТОРІНОК

https://sites.google.com/site/osnovihtml/sposobi

Анотація. *Web-сторінка є текстовим файлом на мові розмітки гіпертексту HTML. Гіпертекст є системою взаємопов'язаних сторінок. Для перегляду Web-сторінок використовують програми-браузери. Для створення Web-сторінок використовують мови HTML і CSS. Існують три основних способи створення Web-сторінок: 1) використання текстового редактора Блокнот; 2) використання спеціальних редакторів документів HTML; 3) використання редактора Word.*

Сучасність відзначається широким розповсюдженням глобальної мережі Internet, базовим протоколом якої, тобто стандартом передавання даних, є WWW (від англ. World Wide Web – світова павутина).

У загальному сенсі "протоколом" називають сукупність правил взаємодії клієнта та сервера.

У свою чергу поняття WWW, тобто світової павутини сторінок, тісно пов'язане з HTML (від англ. Hiper Text Markup Language – мова розмітки гіпертексту).

Гіпертекст є системою взаємопов'язаних сторінок, а гіпертекстове посилання – це невеликий підкреслений і виокремлений іншим кольором текст (картинка чи інший елемент) у документі, натиснувши на якому мишею, отримують доступ до пов'язаного з ним об'єкта, наприклад, тексту, малюнка, музичного або відеофайлу, іншої сторінки тощо.

Web-сторінка є HTML документом, тобто текстовим файлом на мові HTML (розширенням *.htm або *.html), розміщеним у World Wide Web. Web-сторінка, крім тексту, може містити гіпертекстові посилання, за допомогою яких можна переходити до інших Web-сторінок і переглядати їх. Web-сторінка може також містити вставки у вигляді графіки, анімації, відеокліпів і музики.

Для перегляду Web-сторінок використовують програми-браузери: MicroSoft Internet Explorer, Opera, Mozilla Firefox, Google Chrome тощо.

Для створення Web-сторінок використовують мови HTML і CSS. HTML відповідає за структуру і зміст, CSS – за оформлення. Браузер об'єднує HTML і CSS-код і формує зовнішній вигляд сторінки.

У зв'язку з широким розповсюдженням Internet постає питання вивчення основних понять та функціонування сервісів глобальної мережі.

Існують три основних способи створення Web-сторінок (або документів HTML):

1) використання текстового редактора Блокнот (NotePad), убудованого в Windows, і перегляд результатів за допомогою браузера. Цей найпростіший спосіб рекомендується початківцям. Його ми використовуємо як основу методики навчання web-дизайну;

2) використання спеціальних редакторів документів *HTML*, наприклад *Hot Metal Light*, *Hot Dog Professional*, *MS Front Page*, *HTMLPad* тощо;

3) використання редактора *Word* для створення тексту документа, що потім конвертується в *HTML*-формат.

Основні поняття мови *HTML*:

1. Елемент – це конструкція мови *HTML*, або контейнер, що містить дані. Web-сторінка є набором елементів.

2. Тег – це стартовий і кінцевий маркери елемента. Теги визначають границі дії елементів і відокремлюють елементи один від одного. У тексті *Web*-сторінки теги пишуться в кутових дужках, наприклад: <*HTML*>. Кінцевий тег завжди завершується навскісною рискою: </*HTML*>.

Всі теги діляться на парні та одинарні.

Кожний парний тег складається із двох частин: відкритого тегу і закритого. Закритий тег відрізняється від відкритого наявністю символу /.

Приклад парного тегу:

<*p*> Абзац </*p*> У відкритого тегу <p> є пара – закритий тег </*p*>.

Одинарні відкриті теги не мають пари закритого тегу.

3. Гіперпосилання – фрагмент тексту, що є посиланням на інший файл або об'єкт. Гіперпосилання дозволяють переходити від одного документа до іншого.

4. Фрейм – область гіпертекстового документа із власними смугами прокручування.

5. Апплет – програма, передана на комп’ютер клієнта у вигляді окремого файлу, яка запускається при перегляді *Web*-сторінки.

6. Скрипт – програма, включена до складу *Web*-сторінки для розширення її можливостей.

7. Завантаження (*DownLoad*) – копіювання документа з *Web*-сервера на комп’ютер клієнта. *UpLoad* – копіювання документа з комп’ютера клієнта на *Web*-сервер – використовується при створенні власної *Web*-сторінки (тобто при її опублікуванні).

Алгоритм створення web-сторінки:

1) завантажте програму текстового редактора, наприклад, Блокнот;

2) збережіть документ як web-сторінку: «Файл» -> «Зберегти як»: Ім’я файлу – «Ім’я файлу.*html*»; Кодування: *UTF-8*;

3) відкрийте новостворений документ будь-яким браузером;

4) html-код вводимо у вікні блокнота, зберігаючи зміни й оновивлюючи web-сторінку у браузері, – спостерігаємо результат.

Примітка: Головну сторінку сайту прийнято називати *index.html*

Документ HTML має чітко визначену структуру. HTML-документ складається з основного змісту (контенту) і тегів розмітки.

1. Починається з тегу <html> і закінчується відповідним йому тегом - </html>.

2. Містить два розділи, розміщені у строго визначеному порядку: "голова" (заголовок) і "тіло" (зміст).

Заголовок HTML-документу помічається тегами: <head>...</head> - та містить відомості про документ загалом.

Зокрема, він повинен містити теги <title>...</title>, між якими розміщують текст, що повинен відображатися у заголовку вікна браузера.

Крім цього, у розділі заголовків може міститися тег <meta>, призначений для технічного опису документа (ці відомості для пошукових програм), а також тег <style> для опису стилів (наборів параметрів форматування), використаних у документі.

3. Безпосередньо зміст (контент) HTML-документу міститься в "тілі", що розташоване між тегами <body>...</body>.

Отже, в основній структурі HTML-документу обов'язковими є чотири парні теги, які записуються у строго визначеній послідовності.

Розбиття тексту на логічні частини – заголовки.

Мова HTML підтримує шість рівнів заголовків документів, які позначаються відповідно тегами від <h1>...</h1> до <h6>...</h6>. У вікні браузера ці заголовки відображаються різними шрифтами (зазвичай напівжирними).

Хоча в мові HTML є теги форматування, за допомогою яких можна змінювати зображення шрифту, проте використовувати їх для заголовків не рекомендовано.

Текст, що міститься всередині тегу заголовка, відображатиметься відповідно до його рівня.

Найвищий рівень має заголовок <h1>, найнижчий – <h6>.

Використовуючи атрибут align текст заголовка можна вирівняти:

- по центру: <h2 align=center>...</h2>,
- за лівим краєм: <h2 align=left>...</h2> чи
- правим краєм: <h2 align=right>...</h2>.

Для визначення абзаців використовують теги ... . Текст, розміщений між цими тегами, виокремлюється в абзац.

Слід пам'ятати: більше одного пропуску між словами і переходи на новий рядок під час відтворення HTML-документу браузер ігнорує.

При потребі переходу на новий рядок без створення абзацу використовується одинарний тег
.

Розділювачами тексту також можуть бути горизонтальні лінії, які візуально відділяють різні частини документу, - їх створюють за допомогою одинарного тегу <hr>.

Для вирівнювання тексту в абзаці використовується атрибут align=значення (left, right, center, justify).

В "тілі" web-сторінки, що визначається тегом <body>...</body>, розміщують зміст (контент) HTML-документу. Загальне форматування області вмісту та контенту web-сторінки можна здійснити за допомогою таких атрибутів, розміщених у <body>:

- bgcolor=*значення*, де значення – це шестизначний код кольору тла;
- background="значення", де значення – адреса розміщення та назва графічного об'єкта (малюнка) (або лише назва малюнка, при умові, якщо

графічний об'єкт знаходиться в тій же папці, що й HTML-документ) на комп'ютері, або URL – адреса графічного об'єкта, що буде тлом web-сторінки;

- text=*значення*, де значення - це шестизначний код кольору тексту (рис. 10);

- link=*значення*, де значення – це шестизначний код кольору гіперпосилань;

- alink=*значення*, де значення – це шестизначний код кольору гіперпосилань під час натискання;

- vlink=*значення*, де значення – це шестизначний код кольору переглянутих гіперпосилань.

Парний тег ... є контейнером для зміни характеристик шрифту: розміру, кольору та гарнітури.

Атрибут тегу color встановлює колір тексту. При створенні сайту однією із головних проблем є правильна передача кольорів на різних типах комп'ютерів, моніторів і браузерів. Якщо браузер не може правильним чином передати той чи інший колір, то він підбирає схожий або змішує декілька близьких кольорів (dithering). Іноді початковий (зазначений) колір може бути замінений на зовсім невідповідний.

Для забезпечення правильної передачі (без спотворень) кольорів на сайті їх добирають із **палітри "безпечних" кольорів**. Будь-який з 216 кольорів цієї палітри може бути використаний для графіки, тексту і фонів.

Атрибут face визначає гарнітуру шрифту. При встановленні за допомогою атрибуту face гарнітури шрифту варто передбачати, що на комп'ютері користувача, який переглядає web-сторінку, може бути не встановлена ця гарнітура. В такому випадку користувач побачить гарнітуру, встановлену на комп'ютері за замовчуванням.

Для уникнення цього при налаштуванні гарнітури шрифту необхідно вказати два або більше альтернативних шрифти.

Приклад 1: Текст;

Приклад 2: Текст.

Атрибут size задає розмір шрифту в умовних одиницях. Діапазон значень для атрибуту size – від 1 (найменший) – до 7 (найбільший). За замовчуванням встановлюється 2 розмір шрифту.

Хоча тег ... до цих пір підтримується усіма браузерами, проте він вважається застарілим і від його використання поступово відмовляються на користь стилів.

Використання стилів для задання атрибутів шрифту аналогічні тегу ... , проте стилі мають більше можливостей і дозволяють скоротити HTML-код.

Наприклад атрибут назви шрифтів font-family визначає список шрифтів. Приклад реалізації: p {font-family: Arial, Serif}.

Якщо при заданні стилю тексту потрібно використати декілька атрибутів, то їх один від одного відділяють знаком "крапка з комою" (";").

Списки – це позначені маркером взаємопов'язані між собою речення. Списки допомагають структурувати відомості на web-сторінці. Списки є

марковані, нумеровані та список визначень. Відмінні вони за способом оформлення.

Маркований список складається з елементів, кожен з яких позначається спеціальним маркером.

Маркований список формується за допомогою тегу <ul>.... Тип маркеру задається атрибутом type тегу <ul>. А кожний елемент списку розміщується між <li>....

Закритий тег не обов'язковий, проте його варто ставити для чіткого розділення елементів списку.

Закритий тег обов'язковий. Всередині елементу списку можна розміщувати посилання, параграфи, зображення тощо.

Нумерований список теж складається з елементів, проте позначаються вони за допомогою цифр чи букв.

Нумерований список формується за допомогою тегів <ol>.... Кожен елемент списку розміщується між тегами <li>....

Список визначень складається із термінів та їх визначень. Такий список задається за допомогою тегів <dl>...</dl>. Терміни у такому списку вказуються тегом <dt>, а їх визначення – <dd>.

Графічні об'єкти встановлюють на web-сторінку за допомогою одинарного тегу <img src="адреса зображення">. Атрибут src є обов'язковим у тезі <img>.

Наприклад, якщо ви плануєте вставити зображення image.jpg на web-сторінку, то це зображення необхідно зберегти у потрібну папку. Якщо зазначений файл зберегти у ту ж папку, у якій знаходиться сам html-документ, то атрибут src матиме значення: src="image.jpg".

Якщо ж ви плануєте вставити зображення, що знаходиться в Інтернеті за URL-адресою: http://www.glavs.com/userfiles/BrLondon.jpg, то атрибут src матиме значення: src=http://www.glavs.com/userfiles/BrLondon.jpg.

Графічні зображення, які встановлюють на web-сторінку, повинні мати формат, що підтримується браузером. Стандартні формати web-графіки: *.jpg, *.gif, *.png.

Тег <img> також може містити і інші атрибути.

У звичайних текстових редакторах таблиці використовують для наочного подання числової чи текстової інформації. У web-дизайні таблиці відіграють більшу роль. Часто їх використовують для позиціювання графічних чи інших об’єктів на екрані. Такі таблиці утворюють з невидимими межами (рамками), а в клітинках розташовують картинки, тексти тощо.

Теги для створення таблиці <table>… </table>. Теги для створення заголовка таблиці <tc>… </tc>.

Щоб об’єднати у рядку декілька послідовних клітинок, наприклад, дві в одну, у відповідному першому тезі <th> чи <td> записують атрибут rowspan=2.

А щоб об’єднати у стовпці дві клітинки в одну, використовують атрибут colspan=2.

Колір рамки таблиці задають атрибутом bordercolor="колір рамки", а колір тла клітинок – атрибутом bgcolor="колір тла". Товщину рамки в пікселях

задають атрибутом border="товщина рамки". Якщо значення атрибута – число нуль або атрибута немає, то рамка не буде видимою.

Для вдалого розташування таблиць чи рисунків варто проекспериментувати з атрибутами width і height, які задають відповідно ширину і висоту елемента в пікселях чи відсотках до розмірів усього екрану. Наприклад, <table width=300> задає ширину таблиці 300 пікселів; <table width=50%> задає ширину таблиці пів сторінки у горизонтальному напрямку.

Багато web-сторінок для організації посилань використовують одне зображення і до різних його областей прив'язують посилання, утворюючи таким чином "навігаційну карту".

Найрозповсюдженішим прикладом навігаційної карти є географічні (зокрема туристичні) карти світу: натиснувши по частині такої карти, що відповідає певній країні, – відкривається web-сторінка, присвячена цій країні.

Наприклад, для того щоб вставити зображення Knopki.png на web-сторінку і таким чином при натиску на зелену кнопку – перейти на сторінку 1.html, на жовту –2.html, на червону – 3.html потрібно:

1. Позначити рисунок не як звичайне графічне зображення, а як навігаційну карту, присвоївши йому за допомогою атрибуту usemap тегу <img> спеціальне ім'я. Назвемо наше зображення/карту ім'ям panel:

<img src="Knopki.png" usemap="#panel" >.

2. Задати за допомогою тегу <map> із атрибутом name навігаційну карту; для заданого випадку:

<map name="panel">
</map>

3. Визначити області зображення, які будуть посиланнями, та задати шляхи переходів за цими посиланнями. Це здійснюється за допомогою одинарного тегу <area>.

Атрибути тегу <area>:

href – задає до документу, що потрібно відкрити (так само, як у тезі <a>);

shape – визначає форму області зображення, яка буде посиланням; може приймати одне із трьох значень:

rect – прямокутна область;
poly – область у вигляді багатокутника;
circle – область задана колом;
coords – координати.

Налаштовуємо на карті область зеленої прямокутної кнопки як гіперпосилання на web-сторінку "storinka1.html":

Жовта кнопка на зображені Knopki.png має форму трикутника. Для того, щоб виділити її "активну" область на зображені Knopki.png потрібно атрибуту shape присвоїти значення poly – полігон, яке задасть цю область як певний багатокутник, де координатами "X1,Y1,X2,Y2,...,XNYN" – визначатимуться його вершини. Цей багатокутник може бути як правильним, так і неправильним.

Оскільки у заданому випадку вершин всього три, тому координати трикутника необхідно задавати трьома парами значень: "X1,Y1,X2,Y2,X3Y3".

Червона кнопка має форму кола, тому форма "активної" області буде визначатися: shape="circle". Коло задається через: координати центра X,Y та довжину радіуса в пікселях – R. У заданому випадку координати центра кола (X,Y): (483;117), – довжина радіуса R: 57 пікселів.

Контрольні запитання

1. *Що таке протокол та стандарт передавання даних?*
2. *Що таке гіпертекст та гіпертекстове посилання?*
3. *За допомогою чого здійснюють перегляд Web-сторінок?*
4. *Які існують основні способи створення Web-сторінок?*
5. *Назвіть основні поняття мови HTML?*

Практичне(і) завдання

1. *Створіть HTML документ Web-сторінки із використанням одного із трьох основних способів .*

БАЗИ ДАНИХ

Анотація. *База даних це файл у якому зберігається одна або кілька таблиць. Стовпці в таблиці називаються полями, рядки – записами. Поля бувають текстовими, числовими, часовими, логічними та об'єктними. Для зручності роботи з базами даних існують спеціальні програми - системи управління базами даних (СУБД). Однією з популярних СУБД є Microsoft Access.*

У переважній більшості випадків база даних – це файл, в якому зберігається одна або кілька прямокутних таблиць. Стовпці в таблиці також називаються полями, рядки – записами. Поля можуть бути текстовими (Спортивне товариство), числовими (Боксерський ранг), датою та часом (Дата народження), логічними та містити об'єкти (наприклад, зображення, звук, відео). Кількість записів в таблицях реальних баз даних досягає багатьох тисяч.

Для того щоб людина могла зручно працювати з базами даних, написані спеціальні програми – системи управління базами даних (СУБД).

Однією з популярних СУБД є Microsoft Access – програма, що входить в пакет Microsoft Office Professional. У Access можна працювати з реальними базами даних без будь-якого програмування. Також можна програмувати безпосередньо в Access на спеціальній мові Visual Basic for Applications, який є «діалектом» мови Visual Basic по відношенню до пакету Microsoft Office.

Робота з базами даних у програмі Microsoft Access

Спочатку коротко зупинимось на роботі з базами даних в Access, а потім, більш детально, у VB. Матеріал у цьому розділі охоплює деякі необхідні поняття і терміни для того, щоб швидко зрозуміти, що собою являє та чи інша база даних.

Перейдіть до головного меню Microsoft Access → Файл → Створити → Нова база даних → у вікні, збережіть як контакт.mdb. Файл створено...

Додайте таблицю. ... і тут же з'являється вікно бази даних контактів, що пропонує додати таблицю до бази даних.

Двічі клацніть елемент *Створити таблицю в режимі конструктора*. Перед вами з'явиться вікно конструктора таблиць із пропозицією ввести імена та типи полів бази даних.

Створіть і введіть імена полів для таблиці, яку потрібно створити. Биберіть імена без пробілів, щоб було простіше програмувати потім в VB. У другому стовпці виберіть тип даних для кожного поля. Хрестиком закрийте вікно. Програма Access запропонує вам зберегти. Погодьтесь. У вікні, введіть ім'я таблиці – Книги і натисніть кнопку OK. Перед вами з'явиться попередження з питанням про так звані ключові поля (про них трохи пізніше). Дайте відповідь ТАК на підказку створити таке поле. Таблиця створена. На знак цього у вікні бази даних контактів з'явиться значок для даної таблиці.

Заповнення таблиці. Двічі клікніть по значку таблиці і у вікні таблиці можна зручно заповнити таблицю даними. Поле ключа (з іменем Code) буде автоматично заповнено номерами записів (рядків).

При роботі з електронною таблицею Access надає розширене меню і панель інструментів.

Помістивши курсор у поле таблиці, можна відсортувати таблицю за цим полем. Можна друкувати всю електронну таблицю або її частину та виконувати багато інших завдань. Ширину полів можна змінити, перетягнувши межі між іменами полів. Аналогічним чином змініть висоту записів. Ви можете видалити та додати записи, клацнувши правою кнопкою миші лівий сірий стовпець таблиці.

Натиснувши на кнопку дизайнера, можна в будь-який момент перейти до конструктора столів. Тут можна змінити його поля. Клацнувши правою кнопкою миші по лівій сірій колонці дизайнера, можна додавати, видаляти, міняти місцями поля. Щоб повернутися до таблиці, натисніть кнопку таблиці, на яку перетворилася кнопка дизайнера.

Нехай компанія, де працює пара тисяч співробітників, створить базу даних з таблицею з трьох полів: прізвище, ім'я та по батькові співробітника. Напевно серед співробітників знайдуться повні однофамільці, у яких однакові прізвище, ім'я, по батькові. Отже, в базі даних будуть ідентичні записи. І тоді яка користь від такої бази, де не можна відрізнити двох співробітників один від одного? Щоб виправити ситуацію, потрібно ввести ще одне поле, значення якого у всіх неодмінно будуть різними, наприклад, номер паспорта. І було б добре, якби Access подбав про те, щоб значення в цьому полі ніде не повторювалися, адже людина може помилково ввести однакові значення в це поле. Щоб виправити ситуацію, ми повинні зробити поле ключовим. Для цього в конструкторі нам належить натиснути правою кнопкою миші зліва від поля і вибрати з контекстного меню пункт *Ключове поле*. Зліва від поля з'явиться зображення ключика. Поле стало ключовим.

Отже, більшість користувачів бази даних хотіли б, щоб записи в таблиці не дублювалися. Однак для цього не потрібно створювати ключове поле. Так що, очевидно, в таблиці «Книги» навряд чи можна, щоб одна і та ж книга видавалася двічі на день. Це означає, що в таблиці не буде повторюваних записів, навіть якщо немає поля ключа. Ключові поля, однак, відіграють

велику роль в базах даних. Наприклад, вони потрібні програмі Access для роботи зі зв'язаними таблицями. Виходячи з усього цього, бажано додати ключове поле в таблицю «Книги». Для цьогоможна просто відповісти ТАК на розумну пропозицію від Access, в результаті якої було створено ключове поле з назвою Код з типу Лічильник. Це просто номери записів, які автоматично вставляються в тому порядку, в якому вони створені. Ці цифри не можуть бути змінені. У таблиці «Книги» вони знадобляться і в бюрократичних цілях: що таке таблиця без цифр?

Запити до бази даних

Давайте подивимося, як знайти необхідну інформацію в базі даних. Найзручніше для цього використовувати так звані запити.

Простий запит. Припустимо, нам потрібно знайти всі книги товщиною більше 200 сторінок в таблиці «*Книги*». Для цього у вікні *База даних* у списку *Об'єкти* виберіть пункт *Запити*, а потім у режимі конструктора натисніть кнопку *Створити запит*. У вікні запитується, до якого джерела даних звертається запит.

У нас немає інших джерел, окрім таблиці "Книги", тому виберіть її та натисніть кнопку Додати, а потім Закрити. Ми бачимо перед собою вікно конструктора запитів.

У рядку *Поле* вибираємо поля, які потрібно відобразити в запиті. У рядку *Критерії* вказуємо критерій (критерій) для вибору записів у запиті. У нашому випадку пишемо під полем Kol_str умову >200. Закрийте вікно хрестиком. Програма Access запропонує вам зберегти зміни.

У вікні, введіть назву запиту, наприклад, «Товсті книги» і натисніть Ок. Запит створюється. На знак цього в *Запитах* з'явиться піктограма для цього запиту. Двічі клікніть по ньому і ви зможете побачити результати у вікні запиту.

Можна побачити, що після виконання запиту він надає нам інформацію, яку ми шукаємо, в табличній формі. Але запит – це не таблиця. Таблиця – це джерело вихідної інформації, а запит – це зручний перегляд потрібної нам частини цієї інформації, це похідна інформація. Закрийте запит. Поверніться до таблиці та перейдіть до таблиці Книги. Додайте товсту книгу, закрийте таблицю та знову відкрийте запит. Він змінився, включивши нову книгу.

Таким чином, запити автоматично відтворюються при їх відкритті. Пам'ятайте, що під час закриття таблиці вона зберігається автоматично без запиту на збереження. Зверніть увагу, що у вікні запиту можна змінити інформацію, видалити та додати записи так само, як це було у вихідній таблиці. Зміни відразу ж будуть відображені в вихідній таблиці. Це зручно, хоча і змушує бути уважним. За допомогою запиту ви можете перейти до його конструктора і назад, так само, як ви перейшли від таблиці до її конструктора і назад.

Умова відбору може бути складною, включаючи порівняльні знаки, логічні операції, функції Visual Basic для застосунків та назви полів у квадратних дужках.

Умови можуть бути записані одночасно під декількома полями. У цьому випадку вони вважаються пов'язаними логічною операцією *And*. Так, якщо в новому запиті запишемо в поле *Data* умову >#01.01.1960#, а в поле *Kol_str* умову <200#, то цей запит можна назвати «Нові тонкі книги».

У рядку Поле замість імен полів можна писати вирази, наприклад [*Kol_str*]+2. При цьому в стовпці запиту ми побачимо не значення поля, а значення обчислюваного виразу. На таблиці це ніяк не відобразиться.

Крім пунктів «*Таблиці і Запити*» в списку об'єктів у вікні бази даних, є й інші пункти, присвячені важливим аспектам роботи з базами даних. Наприклад, пункт *Звіти* присвячений викладу інформації з таблиць і запитів в зручному для друку вигляді. Але обсяг книги не дозволяє на цьому зупинитися.

Створення бази даних і файлу таблиці у VB

ГлосарійBDE – це набір драйверів, що забезпечують доступ до даних.

Створити глосарій на термінах із тексу з активними посиланнями у інтернеті на їх значення!!!!!

Delphi – одна з найбільш потужних візуально об’єктно-орієнтованих систем програмування, що дозволяє на найсучаснішому рівні створювати як окремі прикладні програми, орієнтовані на роботу в Windows, так і розгалужені комплекси, що призначені для роботи в корпоративних мережах та в Internet.

Object Pascal– мова програмування, в основі якої лежить Pascal, реалізована за допомогою принципів ООП.

Абстракція даних – це можливість визначати нові типи даних, з якими можна працювати майже так само, як із основними.

Алгоритм – система правил, яка описує скінченну послідовність дій, необхідних для розв’язання задачі.

Алгоритмізація задачі – розробка алгоритму її розв’язання. Арифметичний виразявляє собою сукупність констант, змінних, звертань до стандартних функцій і функцій користувача, з’єднаних знаками арифметичних операцій та круглих дужок.

База даних – це спеціальне електронне сховище інформації, доступ до якого здійснюється за допомогою одного або декількох комп’ютерів.

Блок-схема алгоритму – це сукупність, з'єднаних лініями, стандартних блоків, кожен з яких позначає певне правило (дію) алгоритму.

Виконуваний файл – це виконуваний файл додатку;

Вираз – це послідовність знаків операції, операндів та круглих дужок, які в сукупності задають обчислювальний процес для отримання результату.

Віддалені бази даних – це бази даних, які розміщуються на сервері мережі, який також називають віддаленим сервером, а додатки, які працюють з цією базою даних, розміщуються на комп'ютері користувача, що відповідає архітектурі клієнт-сервер.

Відкомпільований файл пакету часу виконання – це пакет часу виконання – файл бібліотеки DLL із певною специфікою Delphi. Візуальнікомпоненти можуть відображатись на формі під час виконання додатку.

Вказівник – це змінна, призначена для збереження адреси в пам'яті.

Впорядкування масиву – послідовне розміщення елементів масиву в комірках пам'яті ПК.

Дескриптор – це є число цілого типу, яке встановлюється для кожного файлу операційною системою.

Динамічна бібліотека – створюється при проектуванні DLL. Динамічний масив – це масив, пам'ять для якого визначається під час виконання програми. Елементи масиву – це дані, які складають масив. Змінні – це елементи програми, що можуть змінювати своє значення у процесі її виконання.

Зовнішня пам'ять– це пам'ять на магнітних дисках. Ідентифікатор – це послідовність літер і цифр, яка починається з літери або знаку підкреслювання.

Індекси – це є комбінація полів таблиці, по яких має проводитись сортування записів таблиці.

Інкапсуляція – це механізм, який об'єднує дані і код, що маніпулює цими даними, а також захищає їх (і дані, і код) від зовнішнього втручання. Інспектор об'єктів є інструментом середовища розробки Delphi. Він дозволяє редагувати власний компонент і визначені для нього обробники подій (реакцію на стандартні події).

Канва (Canvas) – це область компоненти, на якій можна малювати або відображати готові зображення.

Клас – тип даних, який містить поля даних і методи їх обробки (аналог обробки об'єкта в ТР/ВР).

Ключ – комбінація полів набору даних, які однозначно визначають кожен запис в таблиці.

Код– текст програми.

Коментар – це є досить зручний засіб, який використовується для пояснення коду програми або при налагоджуванні програми для тимчасового виключення деяких блоків.

Компонента – клас, ініціалізацію якого можна провести під час розробки форми, який має зручний інтерфейс для зміни своїх властивостей.

Константи – це елементи програми, значення яких не змінюється в процесі її виконання.

Логічний вираз являє собою сукупність констант, змінних, звертань до стандартних функцій і функцій користувача, з'єднаних знаками арифметичних та логічних операцій і круглих дужок.

Локальні бази даних – це бази даних, які розміщуються разом з додатком, що працює з ними, на одному комп'ютері.

Масив – це структура даних, під якою розуміється впорядкована сукупність скінченого числа однорідних даних простої або складної структури, об'єднаних одним іменем.

Методи – це функції та процедури, що забезпечують всі необхідні операції над даними.

Наслідування– це принцип ООП, завдяки якому один об'єкт може набувати властивостей іншого. Невізуальнікомпоненти відображаються на формі тільки на етапі проектування у вигляді піктограм.

Об'єкти проекту – за термінологією візуального програмування – це діалогові вікна і елементи управління, що є в діалогових вікнах (командні кнопки, поля введення текстової інформації, перемикачі та інше).

Об'єктний файл модуля – це відкомпільований файл модуля, який компонується у результуючий виконуваний файл.

Об'єктно-орієнтований підхід – метод, який дає можливість при розв'язанні складних задач в певній предметній області виділити та описати конкретні об'єкти, їх характеристики, зв'язки та взаємодію між ними. Оперативна пам'ять зберігає поточну інформацію (виконувані програми і дані) в даний момент часу, а також ряд допоміжних програм для роботи комп'ютера.

Палітра компонент – це набір (контейнер) компонент Delphi, який являє собою каталог візуальних і прикладних об'єктів.

Панель інструментів – панель, яка містить набір піктограм. Використовується для розташування кнопок, які відповідають деяким командам меню, що часто використовуються.

Підпрограма– це автономна частина програми, яка реалізує певнувідносно самостійну частину загального алгоритму розв'язання задачі і допускає звертання до неї з різних частин програми.

Персональний комп'ютер – невелика за розмірами і вартістю універсальна мікро ЕОМ, призначена для індивідуального користування.

Подія – це те, що відбувається під час роботи Delphi-додатка.

Поліморфізм – це властивість коду вести себе по-різному залежно від ситуації, що виникла в момент виконання. Постійна пам'ять призначена для збереження допоміжної інформації, програм, таблиць тощо.

Предметна область – це частина реального світу, що підлягає вивченню з метою організації керування і у кінцевому рахунку – автоматизації.

Програмне забезпечення – це сукупність програм для управління роботою комп'ютера та обробки інформації, що надходить в комп'ютер.

Проект в Delphi – це розроблюваний Delphi-додаток, що включає сукупність файлів-модулів, файлів-форм та інші файли, які зв'язані власне файлом-проекту.

Реляційна база даних – це є набір пов’язаних одна з одною таблиць.

Розгалуження – алгоритмічна структура, в якій залежно від виконання умови потрібно виконати ту чи іншу дію.

Сервер баз даних – це є програма, яка забезпечує керування віддаленою базою даних та видачу клієнту результатів виконання на надісланий запит.

Середовище візуального об’єктно-орієнтованого програмування Delphi – це графічна автоматизована оболонка над об’єктно-орієнтованою версією мови Pascal – Object Pascal.

Слідування – алгоритмічна структура, в якій кожна наступна дія виконується після завершення попередньої.

Сортування масиву – це процес перегрупування елементів масивуу конкретному порядку.

Список формальних параметрів – це сукупність імен змінних з вказанням їх типу, які служать для передачі даних з програми, яка звертається у функцію.

Статичний масив – це масив, для якого пам’ять виділяється під час його створення.

Текстові файли – це файли, які складаються із символів, що об’єднані в рядки.

Типізований файл – це структурований тип даних, який складається з компонент одного типу і однакової довжини.

Трансляція – переведення програми з мови високого рівня на мову машини. Універсальні мови програмування – це мови програмування високого рівня.

Файл – це сукупність послідовно організованих однотипних даних, які зберігаються на зовнішньому носії інформації.

Файл групи проектів – це файл, що створюється при роботі з групою проектів.

Файл діаграм – це файл діаграми, що створюється на сторінці діаграм Редактора коду.

Файл змісту пакета –це двійковий файл, що містить заголовок пакета та список пакетів, що йому належать.

Файл інформації про пакети – це бінарний файл, що використовується Delphi при роботі із пакетами.

Файл конфігурації – це файл, що зберігає конфігурацію всіх вікон.

Файл модуля – кожній створеній формі відповідає файл модуля, що використовується для збереження коду (можна створювати модулі, які не пов’язані із формами).

Файл опису форми – це файл, який містить інформацію про форму.

Файл пакета – це двійковий файл пакета (package).

Файл параметрів проекту – це файл, в якому зберігаються установки параметрів проекту.

Файл послідовного доступу – це набір даних, в якому до певної інформації можна отримати доступ, послідовно переглянувши дані з його початку до потрібного елемента.

Файл проекту – це текстовий файл, що використовується для збереження інформації про форми та модулі; містить оператори ініціалізації та запуску програми на виконання.

Файл прямого доступу – це набір даних, в якому до потрібної інформації можна отримати доступ безпосередньо, знаючи тільки її розміщення в файлі.

Файл ресурсів – це файл, який містить ресурси, що використовуються проектом.

Файли без типу – це двійкові файли, які можуть містити найрізноманітніші дані у вигляді послідовності байтів.

Файли допомоги – це стандартні файли допомоги Windows, які можуть використовуватися додатком Delphi.

Файли зображень – це файли зазвичай використовуються в додатках.

Файли резервних копій – це відповідно файли резервних копій для файлів проекту, форми та модуля.

Фактичні параметри– параметри, які вказуються при виклику підпрограм-функції.

Форма – це діалогове Windows-вікно програми, яке відкривається під час роботи Delphi-додатку на етапі розробки програми, на яке розміщуються об’єкти проекту.

Фрейм – це панель, тобто деякий фрагмент вікна додатку, який можна переносити на різні форми, в різні додатки, при цьому допускається використання переваг наслідування.

Цикл – багаторазове виконання дії або групи дій.

Бібліографія

Нижче наведено приклад списку праць, цитованих із використанням стилю APA. Упорядковуйте список посилань за алфавітом.

Цитуючи книги, використовуйте формат, показаний у наведених нижче прикладах. Для цього застосовуйте до формату стиль "Бібліографія".

Прізвище автора, перший ініціал або ініціали. (Дата публікації). *Назва книги*. Додаткові відомості. Місто публікації: Видавнича компанія.

Кінг, С. (2000). *Про письменство. Мемуари про ремесло.* Київ: Кишенькові книги.

Посилаючись на онлайнові ресурси, використовуйте формат, показаний у наведених нижче прикладах:

Для документів в Інтернеті

Прізвище автора, перший ініціал або ініціали. (Дата публікації). Назва статті. *Назва праці*. Отримано з: повна URL-адреса джерела

Amir, N. (17 жовтня 2018 р.). 4 поради, як не схибити зі шляху, пишучи твір. *Напишіть документальний твір!* Отримано з: http://writenonfictionnow.com/tips-staying-track-writing/

Для онлайнових періодичних видань

Прізвище автора, перший ініціал або ініціали. (Дата публікації). Назва статті. *Назва періодичного видання*, том і номери сторінок. Отримано з: повна URL-адреса джерела

Brewer, R. L. (4 жовтня 2018 р.). Як краще писати назви: 7 ефективних порад щодо назв для книг, статей і сеансів конференцій. *Дайджест письменника.* Отримано з: http://www.writersdigest.com/whats-new/how-to-write-better-titles

Посилаючись на журнали, використовуйте формат, показаний у наведених нижче прикладах:

Для журналів і періодичних видань

Прізвище автора, перший ініціал або ініціали. (Дата публікації). Назва статті. *Назва періодичного видання, номер тому (номер випуску, якщо можливо*), сторінки.

McPhee, J. (29 квітня 2013 р.). Чернетка № 4. *New Yorker*, *89*, 20-25.

Докладні відомості та вказівки див. в посібнику з публікації APA.

Література та інтернет ресурси:

1. ІвашкоВ. В. Основи програмування: метод. реком. до лабор.практикуму. Чернівці : Чернівецький національний університетімені Юрія Федьковича, 2021. 64 с.
2. https://archer.chnu.edu.ua/xmlui/bitstream/handle/123456789/894/%D0%86%D0%B2%D0%B0%D1%88%D0%BA%D0%BE%20%D0%92%D0%92_%D0%9C%D0%A0%D0%9B%D0%9F%20%D0%B7%20%D0%BA%D1%83%D1%80%D1%81%D1%83%20%D0%9E%D1%81%D0%

BD%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.pdf?sequence=1&isAllowed=y