**13. Змінні середовища Web – сервера. Загальна характеристика.**

При запуску CGI-скрипта веб-сервер передає йому цілий ряд параметрів у змінні середовища. Частина змінних середовища генерується веб-сервером, а інша частина формується з полів HTTP-запиту.

В "Специфікації CGI/1.1" описано стандартний набір змінних середовища CGI, що формуються веб-сервером, а також спосіб передачі CGI-скрипту полів HTTP-запиту в змінних середовища. Крім цього, ряд веб-серверів (включаючи Apache) окрім стандартних формують свої змінні, в котрих передають скрипту додаткові ("нестандартні") параметри (E-Mail адміністратора сервера и т.п.).

**Змінні середовища CGI на конкретному хостингу.**

Щоб побачити всі доступні CGI-скрипту змінні середовища на конкретному сервері, можна запустити на ньому CGI-скрипт, що виведе імена і значення всіх своїх змінних середовища:

**#!/usr/bin/perl**

**print "Content-Type: text/html\n\n";**

**print "<HTML>\n<HEAD></HEAD><BODY>**

**<TABLE width=100% border=1 bordercolor=#00007F cellspacing=0>**

**<TR><TD align=center>Змінна</TD><TD align=center>Значення</TD></TR>\n";**

**foreach $itm(keys %ENV)**

**{print "<TR><TD>$itm</TD><TD>$ENV{$itm}</TD></TR>\n";}**

**print "</TABLE></BODY></HTML>\n";**

Залежно від умов виклику (HTTP-метод, прямий або через SSI...) набір змінних може бути різним.

**Змінні середовища CGI, що формуються веб-сервером.**

**QUERY\_STRING** - рядок параметрів виклику (всі символи, записані в URL після знаку '?').

**REQUEST\_METHOD** - метод HTTP, за допомогою котрого викликаний скрипт. Частіше за все цей метод GET або POST, хоча можуть бути й інші (PUT, DELETE и т.п.).

**GATEWAY\_INTERFACE** - версія інтерфейсу CGI у вигляді CGI/x.y. Наприклад, CGI/1.1

**REMOTE\_ADDR** - містить IP-адресу комп’ютера, з котрого було здійснено звернення до веб-сервера (адреса клієнта або останнього проксі-сервера).

**REMOTE\_PORT** - TCP-порт віддаленого комп’ютера, з котрого йде запит.

**REMOTE\_HOST** - доменне ім’я віддаленого комп’ютера, з котрого йде запит (визначається веб-сервером через DNS по значенню REMOTE\_ADDR, якщо це дозволено його конфігурацією).

**SERVER\_NAME** - доменне ім’я сервера.

**SERVER\_PORT** - номер TCP-порта веб-сервера.

**SERVER\_ADDR** - IP-адреса сервера.

**SERVER\_PROTOCOL** - версія HTTP-протоколу, що використовується для даного HTTP-запиту. Наприклад, HTTP/1.1.

**SERVER\_SOFTWARE** - програмне забезпечення сервера.

**SCRIPT\_NAME** - HTTP-шлях до скрипту.

**SCRIPT\_FILENAME** - фізичний повний шлях до скрипту в файловій системі сервера.

**PATH\_INFO** - HTTP-шлях до скрипту.

**PATH\_TRANSLATED** - повний фізичний шлях до скрипту.

Якщо HTTP-метод, що використовується для методу, передбачає передачу вмісту в тілі запиту, то передаються наступні змінні:

**CONTENT\_TYPE** - тип вмісту (MIME).

**CONTENT\_LENGTH** - довжина вмісту

Якщо стався аутентифікований запит (із вказуванням імені користувача і пароля), то передаються змінні:

**AUTH\_TYPE** - тип аутентифікації (аутентифікаційна схема, що використовується). Частіше за все - 'Basic'.

**REMOTE\_USER** – ім’я користувача, що пройшов аутентифікацію.

Якщо скрипт розрахований на обслуговування декількох користувачів, за параметром **REMOTE\_USER** він може їх розрізняти. Варто мати на увазі, що коли CGI-скрипт уже запущений і йому передано параметр REMOTE\_USER, то користувач ВЖЕ успішно пройшов аутентифікацію на рівні веб-сервера.

**14. Система управління базами даних (СУБД). Характеристика сучасних СУБД. Характеристика MySQL.**

   [База даних](http://lib.mdpu.org.ua/e-book/vstup/glos.htm" \l "%D0%91%D0%B0%D0%B7%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) (БД) - упорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система управління БД.

[Система управління базами даних](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F%20%D0%B1%D0%B0%D0%B7%D0%BE%D1%8E%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) (СУБД) - це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації.

     Централізований характер управління даними в базі даних передбачає необхідність існування деякої особи (групи осіб), на яку покладаються функції адміністрування даними, що зберігаються в базі.

     Головним завданням [БД](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#%D0%91%D0%B0%D0%B7%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин: збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

     Існує величезна кількість різновидів баз даних, що відрізняються за критеріями (наприклад, в Енциклопедії технологій баз даних [21] визначаються понад 50 видів БД).

     Відзначимо тільки основні класифікації.

     Класифікація БД *за моделлю даних*:

* ієрархічні,
* мережеві,
* реляційні,
* об'єктні,
* об'єктно-орієнтовані,
* об'єктно-реляційні.

Класифікація БД *за технологією фізичного зберігання*:

* БД у вторинній пам'яті (традиційні);
* БД в оперативній пам'яті (in-memory databases);
* БД у третинній пам'яті (tertiary databases).

Класифікація БД *за вмістом*:

* географічні.
* історичні.
* наукові.
* мультимедійні.

Класифікація БД *за ступенем розподіленості*:

* централізовані (зосереджені);
* розподілені.

     Окреме місце в теорії та практиці займають просторові (англ. spatial), тимчасові, або темпоральні (temporal) і просторово-часові (spatial-temporal) БД.

*Ієрархічні* бази даних можуть бути представлені як дерево, що складається з об'єктів різних рівнів. Верхній рівень займає один об'єкт, другий - об'єкти другого рівня і т.д.

     Між об'єктами існують зв'язки, кожен об'єкт може включати в себе декілька об'єктів більш низького рівня. Такі об'єкти перебувають у відношенні предка (об'єкт більш близький до кореня) до нащадка (об'єкт більш низького рівня), при цьому можлива ситуація, коли об'єкт-предок не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок. Об'єкти, що мають загального предка, називаються близнюками.

*Мережеві* бази даних подібні до ієрархічних, за винятком того, що в них є покажчики в обох напрямках, які з'єднують споріднену інформацію.

     До основних понять мережевої моделі бази даних відносяться: рівень, елемент (вузол), зв'язок.

     Вузол - це сукупність атрибутів даних, що описують деякий об'єкт. На схемі ієрархічного дерева вузли представляються вершинами графа. У мережній структурі кожен елемент може бути пов'язаний з будь-яким іншим елементом.

     Незважаючи на те, що ця модель вирішує деякі проблеми, пов'язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом.

     Також, оскільки логіка процедури вибірки даних залежить від фізичної організації цих даних, то ця модель не є повністю незалежною від програми. Іншими словами, якщо необхідно змінити структуру даних, то потрібно змінити і додаток.

*Реляційна* модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має наступні властивості:

* кожен елемент таблиці - один елемент даних;
* всі осередки в стовпчику таблиці однорідні, тобто всі елементи в стовпчику мають однаковий тип (числовий, символьний тощо);
* кожен стовпчик має унікальне ім'я;
* однакові рядки в таблиці відсутні;
* порядок проходження рядків і стовпчиків може бути довільним.

*Об'єктна* СУБД ідеально підходить для інтерпретації складних даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого збільшення термінів і вартості розробки додатків. Об'єктна база, на відміну від реляційної, не вимагає модифікації ядра при додаванні нового типу даних. Новий клас і його екземпляри просто надходять у зовнішні структури бази даних. Система управління ними залишається без змін.

*Об'єктно-орієнтована* база даних (ООБД) - база даних, в якій дані оформлені у вигляді моделей об'єктів, що включають прикладні програми, які управляються зовнішніми подіями. Результатом поєднання можливостей (особливостей) баз даних і можливостей об'єктно-орієнтованих мов програмування є об'єктно-орієнтовані системи управління базами даних (ООСУБД). ООСУБД дозволяють працювати з об'єктами баз даних також, як з об'єктами у програмуванні в об'єктно-орієнтованих мовах програмування. ООСУБД розширює мови програмування, прозоро вводячи довготривалі дані, управління паралелізмом, відновлення даних, асоційовані запити й інші можливості.

     Об'єктно-орієнтовані бази даних звичайно рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, які мають складну структуру.

     Система, яка забезпечує об'єктну інфраструктуру і набір реляційних розширювачів, називається "*об'єктно-реляційною*".

     Об'єктно-реляційні системи поєднують переваги сучасних об'єктно-орієнтованих мов програмування з такими властивостями реляційних систем як множинні представлення даних і високорівневі непроцедурні мови запитів.

*За технологією обробки даних* бази даних поділяються на централізовані й розподілені.

*Централізована* база даних зберігається у пам'яті однієї обчислювальної системи. Якщо ця обчислювальна система є компонентом мережі ЕОМ, можливий розподілений доступ до такої бази. Такий спосіб використання баз даних часто застосовують у локальних мережах ПК.

*Розподілена* база даних складається з декількох, можливо пересічних або навіть дублюючих одна одну частин, які зберігаються в різних ЕОМ обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).

*За способом доступу* до даних бази даних поділяються на бази даних з локальним доступом і бази даних з віддаленим (мережевим) доступом.

     Системи централізованих баз даних з мережевим доступом припускають різні архітектури подібних систем:

* файл-сервер;
* клієнт-сервер.

     Файл-сервер. [Архітектура систем](http://lib.mdpu.org.ua/e-book/vstup/glos.htm" \l "%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8) БД з мережевим доступом передбачає виділення однієї з машин мережі в якості центральної (сервер). На такій машині зберігається спільно використовувана централізована БД. Усі інші машини мережі виконують функції робочих станцій, за допомогою яких підтримується доступ користувальницької системи до централізованої бази даних. Файли бази даних відповідно до призначених для користувача запитів передаються на робочі станції, де в основному і проводиться обробка. При великій інтенсивності доступу до одних і тих же даних продуктивність інформаційної системи падає. Користувачі можуть створювати також на робочих станціях локальні БД, які використовуються ними монопольно.

     Клієнт-сервер. У цій концепції мається на увазі, що крім зберігання централізованої бази даних центральна машина (сервер бази даних) повинна забезпечувати виконання основного обсягу обробки даних. Запит на дані, який видається клієнтом (робочою станцією), породжує пошук і вилучення даних на сервері. Витягнуті дані (але не файли) транспортуються по мережі від сервера до клієнта. Специфікою архітектури клієнт-сервер є використання мови запитів SOL.

**MySQL** - вільна система управління базами даних. MySQL є власністю компанії Oracle Corporation, що отримала її разом з поглиненою Sun Microsystems, яка здійснює розробку і підтримку додатку. Розповсюджується під GNU General Public License і під власною комерційною ліцензією, на вибір. Крім цього розробники створюють функціональність на замовлення ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

     Цю систему управління базами даних з відкритим кодом було створено як альтернатива комерційним системам. MySQL із самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем управління базами даних. Вона використовується, у першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

     MySQL є рішенням для малих і середніх додатків. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте до дистрибутиву входить бібліотека внутрішнього сервера, що дозволяє включати MySQL до автономних програм. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNІХ-системах, де є підтримка багатонитевості, що підвищує продуктивність системи в цілому.

     Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі й GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. MySQL характеризується великою швидкістю, стійкістю і простотою використання.

     Для некомерційного використання MySQL є безкоштовною. Можливості сервера MySQL:

* простота у встановленні та використанні;
* підтримується необмежена кількість користувачів, що одночасно працюють із БД;
* кількість рядків у таблицях може досягати 50 млн.;
* висока швидкість виконання команд;
* наявність простої та ефективної системи безпеки.

**15. PHP. Основи синтаксису мови PHP. Прості оператори. Робота з Web-документом.**

РНР - це мова програмування, код якої вбудовується безпосередньо в HTML-сторінку. Програму, написану на РНР, називають РНР-скриптом. При запиті користувача web-cepBep переглядає документ, виконує знайдені в ньому РНР-інструкції (оператори мови РНР), а результат їхнього виконання повертає користувачеві. При цьому статична частина документа, написана мовою HTML, фактично є шаблоном, а змінювана частина формується нри виконанні РНР-інструкцій. Для віддаленого користувача подібні документи нічим не відрізняються від звичаіїних статичних HTML-докумеитів, за винятком того, що в розширенні імені фаіїлу для таких документів може стояти не htm або html, а php (а в старих версіях phtml або php3). РНР - це система розробки скриптів, що включає в себе CGI - інтерфейс, інтерпретатор мови та набір функцій для доступу до баз даних і різних об'єктів WWW (функції для роботи з електронною поштою, FTP, http, тощо). На сьогодні, РНР є одним із найбільш зручних і водночас достатньо потужним засобом розробки додатків WWW і інтерфейсів до баз даних в мережі Інтернет. РНР-скрипти знаходяться на сервері і їхній вміст відвідувачеві сайту переглянути неможливо. Файли скриптів мають розширення \*.php. При активації скринту (заниті користувача) серверна програма виконує всі команди php цього скринту, не торкаючись статичної частини документа (HTML-код) і результат повертається програмі-браузеру. Після виконання РНР-скринта користувач бачить звичайну веб-сторінку, яка відрізняється від інших тільки розширенням.

**Установка РНР для роботи із сервером Apache**

Можлива установка РНР як модуля Apache і як окремого CGI-модуля.

Для установки необхідно:

* зупинити сервер Apache;
* розпакувати файли РНР у який-небудь каталог (наприклад,*c:\php\);*
* скопіювати файл*php4ts.dll* у*windows\system(32)\;*
* змінити настроювання Apache у файлі*apache\conf\httpd.conf:*

а) при установці як модуля:

*LoadModulephp4\_module c:/php/sapi/php4apache.dll*

*AddType application/x-httpd-php .php4*

*AddType application/x-httpd-php .phtml*

*AddType application/x-httpd-php .php3*

*AddType application/x-httpd-php .php*

*AddType application/x-httpd-php-source .phps*

*# Додаємо PHP - розширення в рядок*

*Directorylndex index.html index.phtml index.phpS default.php default.phpS default.phtml*

б) при установці як CGI - додатка:

*ScriptAlias /php/ "C.Vphp/"*

*<Directory "C:/php">*

*AllowOverride None*

*Options None*

*</Directory>*

*Action application/x-httpd-php "/php/php.exe"*

*AddType application/x-httpd-php .phtml*

*AddType application/x-httpd-php .php3*

*AddType application/x-httpd-php .php*

*Action application/x-httpd-php-source "/php/php.exe"*

*AddType application/x-httpd-php-source .phps*

*# Додаємо PHP - розширення в рядок:*

*Directorylndex index.html index.phtml index.phpS default.php default.php3 default.phtml*

* · скопіювати файл*php3-dist.ini* у каталог*\\*windows\*\,* перейменуйте його в *php.ini*
* · роз коментувати у файлі*php.ini* рядок:*;extention=iм'я\_мoдyля.dll*для тих модулів, що планується використовувати (для роботи з MySQL завантаження модулів не потрібно - її підтримка убудована в РНР4).

**Вступ у PHP**

PHP - це скрипт-мова (scripting language), що вбудовується в HTML, що інтерпретується й виконується на сервері. Простіше всього це показати на прикладі.

Створіть у каталозі сервера файл*ex01.php,* що містить наступний текст.

*<html>*

*<head>*

*<title>Example</title>*

*</head>*

*<body>*

*<? echo "Hi, I'm a PHP script!"; ?>*

*</body>*

*</html>*

Потім запустивши браузер, наберіть у рядку URL: *http://server\_url/labal/ex01.php,* де server\_url - адреса вашого web-cepвepa. Після виконання цього скрипту ми одержимо сторінку.

Розглянемо даний приклад докладніше: усі рядки приклада крім однієї – це звичайний статичний набір тегів HTML. Найбільший інтерес представляє рядок, укладений у тег <? ?>. Це і є РНР-скрипт. У даному випадку - це єдина команда РНР -*echo,* що здійснює вивід у потік стандартного висновку фрази 'Ні, I’m а РНР script!'. Таким чином, після обробки сторінки на сервері браузерові клієнта передається наступний HTML-код:

*<html>*

*<head>*

*<title>Example</title>*

*</head>*

*<body>*

*Hi, I'm a PHP script!"*

*</body>*

*</html>*

фактично, після обробки сервером, замість блоків коду РНР підставляються результати їхнього виконання, тобто те, що дана ділянка коду вивела в стандартний потік виводу (наприклад, функцією*echo).*

Основна відмінність РНР від CGI-скриптів, написаних на інших мовах, типу Perl чи С - це те, що в CGI-програмах ви самі пишете виведений HTML-код, а, використовуючи РНР - ви вбудовуєте свою програму в готову HTML-сторінку, використовуючи відкриваючий і закриваючий теги <? і ?>.

Відмінність РНР від JavaScript, полягає в тому, що РНР-скрипт виконується на сервері, а клієнту передається результат роботи, тоді як у JavaScript-код цілком передається на клієнтську машину, і тільки там виконується.

РНР дуже схожий на Active Server Pages (ASP) і Java Server Pages (JSP). Усі три мови дозволяють розміщати код, який виконується на Web-cepвepi, всередині HTML сторінок.

**Можливості РНР**

У декількох словах: на РНР можна зробити все, що можна зробити за допомогою CGI-програм. Наприклад: обробляти дані з форм, генерувати динамічні сторінки, одержувати й посилати куки (cookies).

Крім цього в РНР включена підтримка багатьох баз даних (databases), що робить написання Web-програм із використанням БД дуже простим.

От неповний перелік підтримуваних БД:

Dabase,        InterBase,       Solid;

dBase,          MSQL,            Sybase;

Empress,      MySQL,          Velocis;

FilePro,        Oracle,            Unix rdbm;

Informix,      PostgreSQL.

В добавок до всього РНР розуміє протоколи ІМАР, SNMP, NNTP, РОРЗ і навіть HTTP, а також має можливість працювати із сокетами (sockets) TCP/IP і спілкуватися на інших протоколах. Синтаксис РНР дуже схожий на синтаксис С чи Perl. Люди, знайомі з програмуванням, дуже швидко зможуть почати писати програми на РНР. У цій мові немає строгої типізації даних і немає необхідності в діях по виділенню/звільненню пам'яті.

Програми, написані на РНР, досить легкочитаємі. Написаний РНР – код легко прочитати й зрозуміти, на відміну від Perl-програм.

**Синтаксис РНР**

Скрипт мова РНР подібна по синтаксисі мові С за багатьма показниками. Вона підтримує змінні, масиви, звертання до функцій, різні типи змінних і безліч інших речей, що вам можуть знадобитися для написання складних програм. Кожна команда РНР починається з тега <?, і закінчується ?>. Команди можуть бути згруповані усередині однієї пари <? ?> і відокремлюватися друг від друга символом ; . Підтримуються змінні, імена змінних починаються із символу $. Так, наприклад, щоб привласнити значення 5 змінної а, потім відобразити її, можна написати наступний фрагмент:

*<?$а = 5.?>*

*<?echo $а?>*

це можна записати також у виді:

*<?; $а = 5; echo $а ?>*

чи навіть:

*<?$а* = 5;

*echo $а;?>*

це можна побачити у прикладі*ex02.php.*

**Приклад***ex02.php*

*Перший тип об'яви: <br>*

*<?$а***= 5?>**

*<?echo $а?> <br>*

*Другий тип об'яви: <br>*

*<?; $а = 5; echo $а ?> <br>*

*Третій тип об'яви: <br>*

*<? $а= 5;*

*echo $а; ?>*

Зайві символи пробілу, табуляції і нового рядка ігноруються. Це потрібно для того, щоб формувати блоки програми РНР, для більшої зручності читання. Регістр написання має значення для імен змінних, але не для імен функцій. Пізніше в цій документації, при огляді функцій, регістр використовується тільки для того, щоб зробити імена функцій більш читабельними. У фактичній програмі Ви можете використовувати будь-який регістр, що побажаєте. Коментарі підтримуються. Коментарі записуються точно так само як коментарі в Мові С. /\* починає коментар, \*/ закінчує коментар. Коментарі можуть бути поміщені в будь-якім місці усередині блоку <? ... ?>.

**16. PHP. Типи даних мови PHP. Приведення типів. Визначення змінних.**

Імена змінних позначаються знаком $ (долара). Те ж саме "Привіт, я - скрипт PHP!" Можна отримати наступним чином:

<?php

$message = "Привіт, я - скрипт PHP!" ;

echo $message ;

?>

**Типи даних в PHP**  
PHP підтримує вісім простих типів даних: Чотири скалярних типи:  
-boolean (двійкові дані)  
-integer (цілі числа)  
-float (числа з плаваючою крапкою або 'double')  
-string (рядки)  
Два змішаних типи:  
-array (масиви)  
-object (об'єкти)  
І два спеціальних типи:  
-resource (ресурси)  
-NULL ("порожні")  
Існують також кілька псевдотипів:  
-mixed (змішані)  
-number (числа)  
-callback (зворотного виклику)

Опис типу змінної не є обов'язковим і тому програмістом може бути і не визначений. Кожна змінна автоматично перетворюється в кожній з типів й різні функції використовують потрібний тип. проте, існує декілька функцій, для яких важливий тип змінної. Для ініціалізації (визначення) змінної необхідно їй просто присвоїти значення.

<?php

$а = 5; - змушує змінну $а стати змінною типу Integer

$b = 5.0; - змушує змінну $b стати змінною типу Double

$0 = "5"; - змушує змінну $с стати змінною типу String

?>

Зазвичай використовують три типи змінних. Довгі цілі (long int), подвійної точності з плаваючою комою, (double) і символьні рядки (strings). Тип змінних виявляється автоматично. Наприклад:

*<?$а = 5?> -* змушує*$а* стати змінної*типу INTEGER;*

*<?$а = 5.0?> -* змушує*$а* стати змінної типу*DOUBLE;*

*<?$а = "5"?> -* змушує*$а* стати змінної типу*STRING.*

Тип змінної взагалі не дуже важливий. Кожна змінна, незалежно від типу, перетворюється в кожній із трьох типів внутрішньо, і різні функції пробують використовувати правильний тип. Є тільки кілька функцій, для яких важливий тип змінної. Усі три типи змінних можуть також розглядатися як масиви, якщо до їхніх імен додається [значення]. На відміну від С, масиви в РНР фактично являють собою асоціативні масиви, подібні тим, що використовуються в Perl. Наступний запис, вірний:

*<? $а[0] = 5;*

*$а ["hello"] = 6;*

*echo $a[0];*

*echo $a["hello"]; ?>*

зверніть увагу, що, якщо ім'я змінної використовується, і як масив і як звичайна змінна, то ім'я звичайної змінної є синонімом елементу масиву з індексом «0». Тобто*$а = 1,* це теж саме, що*$а[0] = 1.*

РНР також підтримує не-індексовані масиви. Не-індексований масив генерує власний індекс, у міру додавання елементів до нього. Наприклад,:

*$а[] = "Hello";*

*$а[] = "There";*

першому елементу, що вставляється в не-індексований масив, завжди привласнюється індекс 0, другий 1 індекс, і m.д. Отже вищезгадані елементи можуть бути роздруковані за допомогою:

*echo $а[0];*

*echo $а[1];.*

Ви можете використовувати функцію count(), для того щоб визначити кількість елементів для будь-якого масиву.

Ще одна властивість мови, це те, що тип змінної визначає, які основні операції можуть бути виконані. Наприклад,*$а = $b + $с*може поводитися подвійно. Якщо $b це число, то числове значення $с додається до $b, і сума зберігається в $а. У цьому випадку тип $с не важливий. Операція керується типом першої змінної. Якщо $b рядок, то значення рядка $с конкатенується з $b, і результуючий рядок міститься в $а. Це також приводить до деяких непорозумінь.

Приклад*ex03.php:*

*<? $а[] = "Hello";*

*$а[] = "There";*

*echo $а[0];*

*echo "<br>";*

*echo $a[l];*

*echo "<br> кількість елементів масиву a : ";*

*echo count($a)?>*

При використанні змінних тину string можна використовувати як подвійні так і одинарні лапки. Але між ними є різниця :

**<?**

$оріІІа='Оболонь';

$8='Пиво $орІІІа';

echo $s;

//виводить Пиво $оріІІа

echo '<br>';

$8="Пиво $оріІІа";

echo $s:

// виводить Пиво Оболонь

**?>**

Змінна розглядається як масив, якщо до її імені додається значення індекса, взятого у квадратні дужки []. Наприклад

<?php

$а[0] = 10;

$а[1] = 14;

$а[2] = 8;

//Дістали масив $а з трьох елементів

?>

Індексом масиву може бути або ціле число (Integer) або стрічка (string), елементами масиву можуть бути значення будь-якого типу.

**<?**

$a['name']='shoues';

$a['color']=Yed';

$a['size']=36;

**?>**

Масив, в якому індексами є стрічки називається асоціативним.

**17. PHP. Вирази і операції. Вирази і операції мови PHP. Оператори мови PHP.**

Операції, що визначені в мові php

**Арифметичні операції**

|  |  |  |
| --- | --- | --- |
| **Приклад** | **Назва** | **Результат** |
| $а + $b | Додавання | Сума $а і $b |
| $а - $b | Віднімання | Різниця $а і $b |
| $а \* $b | Множення | Добуток $а і $b |
| $а / $b | Ділення | Частка від ділення $а на $b |
| $а % $b | Остача | Остача від цілочисельного ділення $а на $b |

**Логічні операції**

|  |  |  |
| --- | --- | --- |
| **Приклад** | **Назва** | **Результат** |
| *$а == $b* | дорівнює | **TRUE,** якщо $а дорівнює $b. |
| *$а === $b* | ідентично | **TRUE,** якщо $а дорівнює  $b  і вони одного типу, (тільки в РНР4**)** |
| *$а != $b* | не дорівнює | **TRUE,** якщо $а не дорівнює $b. |
| *$а о $b* | не дорівнює | **TRUE,**  якщо  $а  не дорівнює $b. |
| *$а !== $b* | не ідентично | **TRUE,**  якщо  $а  не дорівнює $b або вони різних типів, (тільки в РНР 4 і більше) |
| *$а < $b* | менше | **TRUE,** якщо $а строго  менше $b. |
| *$а > $b* | більше | **TRUE,** якщо $а строго більше $b. |
| *$а <= $b* | менше або дорівнює | **TRUE,** якщо $а менше або дорівнює $b. |
| *$а >= $b* | більше або дорівнює | **TRUE,** якщо $а більше або дорівнює $b. |
| *! $а* | Заперечення | **TRUE,**  якщо  $а  не **TRUE.** |
| *$ а & &* | Логічне «і» | **TRUE,** якщо і $а, і $b **TRUE.** |
| *$а \\ $b* | Логічне «або» | **TRUE,** якщо $a або $b **TRUE.** |

**Стрічкові операції**

Основними операціями із стрічками є:

* конкатенація  - повертає об'єднання із правого та лівого аргументів.
* присвоєння ('.=') - приєднує правий аргумент до лівого аргументу.

**<?**

$а = "Hello

$b = $а . "World!": // тепер $b містить "Hello World!"

$а = "Hello ";

$а .= "World!"; // тепер $а містить "Hello World!"

**?>**

Стрічку можна розглядати і як масив символів

<?

$а='НеІІо';

$с=$а[1];

// с буде мати значення 'е'

**?>**

*Нумерація символів, як і в масивах, починається з нуля.*

**Операції присвоєння**

Крім базової операції присвоєння (=), існують "комбіновані операції" для всіх бінарних, арифметичних І стрічкових операцій, які дають змогу використати значення у виразі, а потім установити його значення в результат цього виразу. Наприклад:

$3 — 3j

$а += 5; // установлює в $а 8,

//ніби ми сказали: $а = $а + 5;

$b = "Hello

$b .= "There!"; // встановлює в $b "Hello There!",

// аналогічно $b = $b . "There!";

**Операції інкременту/декременту**

PHP підтримує операції pre- і post-інкременту й декременту.

|  |  |  |
| --- | --- | --- |
| **Приклад** | **Назва** | **Ефект** |
| ++$а | Pre-increment | Збільшує $а на 1, потім повертає $а. |
| $а++ | Post- | Повертає $а, потім збільшує $а на 1. |
| ~$а | Pre-decrement | Зменшує $а на 1, потім повертає $а. |
| $а— | Post- | Повертає $а, потім зменшує $а па 1. |

Наведемо приклад скрипту, який демонструє дію цих операцій:

<?php

echo "<h3>Postincrement</h3>";

$а = 5;

echo "Повинне бути 5:". $а++ . "<br> ";

echo "Повинне бути 6:". $а . "<br>";

echo "<h3>PreincreiTient</h3>";

$а = 5;

echo "Повинне бути 6:". ++$а . "<br>";

echo "Повинне бути 6:". $а . "<br>";

echo "<h3>Postdecrement</h3>";

$а = 5;

echo "Повинне бути 5:". $а- . "<br>";

echo "Повинне бути 4:". $а . "<br>";

echo "<h3>Predecreinent</h3>";

$а = 5;

echo "Повинне бути 4:". -$а . "<br>";

echo "Повинне бути 4:". $а . "<br>";

**?>**

**Математичні вирази**

PHP підтримує повний набір математичних операцій, у виразах. Враховується порядок операцій. Наступні оператори, припустимі:

*<? $а = 2 + 1 ?> Додавання*

*<? $а = 2 -1 ?> Віднімання*

*<? $а = 2 \* 1 ?> Множення*

*<? $а = 2/1 ?> Ділення*

*<? $а = 2 % 1 ?> Ділення по модулю*

підтримуються і дужки і порядок операцій, так, що наступна запис вірний:

*<?$а = (2+1)''3+6/3?>*

підтримуються С-подібні оператори збільшення += і зменшення -= . Тобто

*<? $а +— $b?>*

це еквівалентно:

*<? $а = $а + $b?>*

підтримуються С-подібні порозрядні оператори =& і |=. Тобто.

*<? $а&= 4?>*

це еквівалентно:

*<? $а = $а & 4?>.*

**Список деяких математичних функцій php**

**abs -** абсолютне значення

**acos -** арккосинус

**asin -** арксинус

**atan -** арктангенс

**cos -** косинус

**cosh -** гіперболічний косинус

**deglrad -** конвертує число в градусах в еквівалент у радіанах

**ехр -** е в степені...

**log10 -** логарифм із основою**10**

**log -** натуральний логарифм

**max($argl,$arg2,...) -** повертає найбільше значення з значень параметрів Sargl, $arg2 ...

**pi -** значення pi

**pow(base,exp) -** Повертає*base,* в степені*ехр.*

**rad2deg -** конвертує число в радіанах в еквівалент у градусах

**rand -** генерує випадкове число

**round -** округляє число із плаваючою крапкою/АоаІ

**sin -** синус

**sinh -** гіперболічний синус

**sqrt -** квадратний корінь

**tan -** тангенс

**tanh -** гіперболічний тангенс

**18. PHP. Оператори розгалуження, циклу, виклик функцій.**

Умовні оператори if, else, elseif, switch

Конструкція **if**  
Вказані дії виконуються тільки тоді, коли умова є істинною

if (умова) { if($index > 0){

Дія; echo 'Index > 0';

} }

Конструкція **if...else**  
Якщо умова істинна, виконується дія із блоку if, в протилежному випадку — з блоку else.

if(умова){ if($index > 0){

Дія; echo ‘Так';

}else{ } else {

Дія; echo 'Ні';

} }

Конструкція **elseif**  
Якщо умова блоку if істина, виконаються дії блоку if. В іншому випадку, якщо умова блоку elseif істина, виконаються дії блоку elseif. У всіх інших випадках виконається дії з блоку else.

if(умова){ if($numb= 5 &&

$numb = 10)

Дія; $discount = 5;

}else{ }else{

Дія; $discount =10;

} }

Конструкція **switch** Якщо значення змінної відповідає значенню одного з блоків case, виконаються дії з цього блоку. В іншому випадку - з блоку default.

switch(Змінна){

case Значення 1:

Дія 1;

[break;]

case Значенння 2:

Дія 2;

[break;]

[default: Дія;]

}

switch($day){

case 1:

echo 'Понеділок';break;

case 2:

echo 'Вівторок'; break;

case 3:

echo 'Середа'; break;

case 4:

echo 'Четвер'; break;

case 5:

echo 'П’ятниця'; break;

case 6:

echo 'Субота'; break;

case 7:

echo ‘Неділя'; break;

default:

echo 'Немає такого дня';

Оператори циклу for, while

Цикли призначені для багаторазового виконання набору інструкцій. У **циклі for**вказується початкове і кінцеве значення лічильника, а также крок, з яким лічильник буде змінюватися. Змінюватися лічильник може як в позитивну, так і негативну сторону. Дії виконаються стільки разів, скільки ітерацій пройде від початкового значення лічильника до досягнення кінцевого, з вказаним кроком.

for(початок;кінець;крок){ for($i = 1; $i = 5; $i++) {

Дія; $sum += $i;

... eho $sum;

} }

**Цикл while.** Дії будуть виконуватися до тих пір, поки умова істинна. **Цикл while**є циклом з передумовою.

while (умова) { while ($state == ‘Ранок') {

Дія; echo 'Робочий день починається';

... $state = ‘Обід';

} }

**Цикл do ... while** Цикл **do ... while** є циклом з післяумовою. Це означає, що спочатку буде виконуватися дія, а потім перевірятися умова. Таким чином дія завжди виконається мінімум один раз.

do{ do{

Дія; echo ‘Удар';

... } while ($state == ‘Бокс');

} while (умова);

Оператори управління циклами break, continue

**Break** перериває роботу циклу. Інтерпретатор перейде до виконання інструкцій, наступних за циклом. **Continue**перериває виконання поточної ітерації циклу. Цикл продовжить виконуватися з наступної ітерації

$index = 1;

while ($index < 10){

echo "$index ";

$index++;

if ($index == 5)

break;

}

$index = 0;

while ($index < 10){

$index++;

if ($index == 5)

continue;

echo "$index";

}

Оператор циклу foreach

**Цикл foreach**Дуже зручний при роботі з масивами. Зазначені дії виконуються для кожного елемента масиву $array, при цьому $key - номер елемента масиву $array, $ value - значення цього елемента.

foreach ($array as [ $key => ] $value){

Дія;

...

}

$pets[] = 'Собака';

$pets[] = 'Кіт';

$pets[] = 'Риба';

foreach ($pets as $index => $value) {

echo "Елемент № $index має значення: \"$value\"<br>";

}

Приклад 2

<html>

<head>

<title>Перегляд асоціативного масиву</title>

</head>

<body>

<?php

$ціна = array ("помідори" => 15, "огірки" => 12);

foreach ($ціна as $овочі => $грн)

{

echo "$овочі коштують $грн грн.<br>";

}

?>

</body>

</html>

РЕЗУЛЬТАТ ПРИКЛАДУ 2:  
помідори коштують 15 грн. огірки коштують 12 грн.  
  
ПРИКЛАД 3

<html>

<head>

<title>Перегляд двовимірного масиву</title>

</head>

<body>

<?php

$дані = array (

"Іванов" => array ("ріст" => 174, "вага" => 68)

"Петров" => array ("ріст" => 181, "вага" => 90),

"Сидоров" => array ("ріст" => 166, "вага" => 73));

foreach ($дані as $прізвище => $дані1)

{

echo "<br>$прізвище:<br>";

foreach ($дані1 as $параметр => $pp)

{

echo "$параметр = $pp<br>";

}

}

?>

</body>

</html>

РЕЗУЛЬТАТ ПРИКЛАДУ3 :  
Іванов: ріст = 174 вага = 68  
Петров: ріст = 181 вага = 90  
Сидоров: ріст = 166 вага = 73  
  
ПРИКЛАД 4

<?php

$a = array ('a' => 'apple ', 'b' => 'banana ',

'c' => array ('x ',' y ',' z'));

print\_r ($a);

?>

РЕЗУЛЬТАТ ПРИКЛАДУ 4 :  
Array ( [a] => apple [b] => banana [c] => Array ( [0] => x [1] => y [2] => z ))  
print\_r - Вивід усіх эллементів масиву. Зручна функція для відладки.

Підрахунок кількості елементів

Кількість елементів в масиві можна визначити за допомогою функцій count() або sizeof().

<html>

<head>

<title>Розмір масиву</title>

</head>

<body>

<?php

$фрукти = array("яблуко", "груша", "слива", "персик");

echo "Розмір масиву \$фрукти рівний ".count($фрукти)".<br>";

echo "Останній елемент масиву \$фрукти - ".$фрукти[count($фрукти) - 1]".<br>";

?>

</body>

</html>

Частота входження елементів в масив

Частоту входження елементів в масив можна визначити за допомогою функції array\_count\_values \_. Ця функція повертає масив, в якому ключами є елементи досліджуваного масиву, і значеннями - частоти їх входження в досліджуваний масив.

>html>

>head>

>title>Розмір масиву>/title>

>/head>

>body>

>?php

$фрукти = array("яблуко", "груша", "слива", "персик", "груша");

print\_r (array\_count\_values($фрукти));

?>

>/body>

>/html>

РЕЗУЛЬТАТ ПРИКЛАДУ 5:   
Array ( [яблуко] => 1 [груша] => 2 [слива] => 1 [персик] => 1 )  
Функція print\_r() відображає ключі і значення масиву, вказаного в аргументі.

PHP підтримує концепцію змінних функцій. Це означає, що коли до імені змінної прикріплені круглі дужки, PHP шукає функцію з тим же ім’ям, що і результат обчислення змінної, і намагається її виконати. Цю можливість можна використовувати для реалізації обернених викликів, таблиць функцій і багатьох інших речей.

Змінні функції не будуть працювати з такими мовними конструкціями як [echo](https://www.php.net/manual/ru/function.echo.php), [print](https://www.php.net/manual/ru/function.print.php), [unset()](https://www.php.net/manual/ru/function.unset.php), [isset()](https://www.php.net/manual/ru/function.isset.php),[empty()](https://www.php.net/manual/ru/function.empty.php), [include](https://www.php.net/manual/ru/function.include.php), [require](https://www.php.net/manual/ru/function.require.php) і т.п. Вам необхідно реалізувати свою функцію-обгортку для того, щоб наведені вище конструкції могли працювати з змінними функціями.

Приклад. Робота з функціями за використанням змінних

<?php  
function foo() {  
    echo "В foo()<br />\n";  
}  
  
function bar($arg = '')  
{  
    echo "В bar(); аргумент був '$arg'.<br />\n";  
}  
  
// Функція-обгортка для echo  
function echoit($string)  
{  
    echo $string;  
}  
  
$func = 'foo';  
$func();        // Викликає функцію foo()  
  
$func = 'bar';  
$func('test');  // Викликає функцію bar()  
  
$func = 'echoit';  
$func('test');  // Викликає echoit()  
?>