Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №2 по дисциплине

ТЕХНОЛОГИИ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ

на тему

«Ветвления и функции»

|  |  |
| --- | --- |
| Выполнил ст. гр. 820601  Проверил преп. каф. ИТАС | А.С. Хасан  А.Л. Гончаревич |

Минск 2022

СОДЕРЖАНИЕ

[Введение 3](#_Toc94186760)

[1 Постановка задачи 4](#_Toc94186761)

[2 Теоритическая часть 5](#_Toc94186762)

[3 Ход работы 7](#_Toc94186763)

[Заключение 11](#_Toc94186764)

# ВВЕДЕНИЕ

Протокол передачи гипертекста (*Hypertext Transfer Protocol* — *HTTP*) —это прикладной протокол для передачи гипертекстовых документов, таких как *HTML*. Он создан для связи между веб-браузерами и веб-серверами, хотя в принципе *HTTP* может использоваться и для других целей. Протокол следует классической клиент-серверной модели, когда клиент открывает соединение для создания запроса, а затем ждёт ответа. *HTTP* — это протокол без сохранения состояния, то есть сервер не сохраняет никаких данных (состояние) между двумя парами «запрос-ответ». Несмотря на то, что *HTTP* основан на *TCP*/*IP*, он также может использовать любой другой протокол транспортного уровня с гарантированной доставкой.

*PHP* (рекурсивный акроним словосочетания *PHP*: *Hypertext Preprocessor*) — это распространённый язык программирования общего назначения с открытым исходным кодом. *PHP* специально сконструирован для веб-разработок и его код может внедряться непосредственно в *HTML*.

# ПОСТАНОВКА ЗАДАЧИ

Ознакомиться с основами ветвления программы, функциями, областью видимости и времени жизни переменных.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Ход выполнения программы может быть линейным, то есть таким, когда выражения выполняются друг за другом. Ни одна строка кода программы не пропускается.

Однако чаще в программах бывает не так. При выполнении кода, в зависимости от тех или иных условий, некоторые его участки могут быть опущены, в то время как другие – выполнены. Иными словами, в программе может присутствовать ветвление, которое реализуется условным оператором – особой конструкцией языка программирования.

Конструкция *if* является одной из наиболее важных во многих языках программирования, в том числе и *PHP*. Она предоставляет возможность условного выполнения фрагментов кода. Структура *if* реализована в *PHP* по аналогии с языком *C*:

*if* (выражение)

{

инструкция;

}

Инструкции *if* могут быть бесконечно вложены в другие инструкции *if*, что даёт большую гибкость в организации условного выполнения различных частей программы.

Ещё одним условным оператором является тернарный оператор «?:». Выражение (*expr*1) ? (*expr*2) : (*expr*3) интерпретируется как *expr*2, если *expr*1 имеет значение *true*, или как *expr*3, если *expr*1 имеет значение *false*.

Оператор *switch* похож на ряд операторов *if* с одинаковым условием. Во многих случаях может понадобиться сравнивать одну и ту же переменную (или выражение) с множеством различных значений и выполнять различные участки кода в зависимости от того, какое значение принимает эта переменная (или выражение). Это именно тот случай, для которого удобен оператор *switch*. Структура *switch* в *PHP*:

*switch (exp)*

*{*

*case condition1:*

*exp1;*

*break;*

*case condition2:*

*exp2;*

*break;*

*default:*

*exp4;*

*break;*

*}*

Сначала записывается ключевое слово *switch*, после которого в скобках записывается некоторое выражение. Далее, после слова *case* нужно перечислить возможные варианты значений, если значение истина, то выполняется группа операторов, которые записаны до оператора *break*. Если ни одно условие не подходит, то выполняется оператор *default* (если оператор *default* не записывать, то при невыполнении никаких других условий ничего не произойдет).

Функции представляют собой блок инструкций, которые многократно можно вызывать в различных частях программы. Функции позволяют разделять программу на меньшие функциональные части. Общий синтаксис определения функции выглядит следующим образом:

*function* имя\_функции([параметр [, ...]])

{

// инструкции

}

Переменные по области видимости можно разделить на глобальные и локальные. К глобальным можно обращаться из любого места программы, локальные доступны только в конкретном ее месте. Все переменные, которые объявлены в теле скрипта вне функций, – глобальные. Переменные, объявленные в функциях, имеют локальную область видимости. Переменные, которые функция принимает в качестве параметров, также являются локальными.

# 3 ХОД РАБОТЫ

Работа выполняется с помощью редактора кода – *Visual Studio Code*. Работа с программой начинается с создания *PHP* файла, в который будут помещаться скрипты.

В задании 2.1 необходимо было объявить две целочисленные переменные *a* и *b* и задать им произвольные значения, затем написать скрипт, который работает по следующему принципу:

* если *a* и *b* положительные, вывести их разность;
* если *а* и *b* отрицательные, вывести их произведение;
* если *а* и *b* разных знаков, вывести их сумму.

Ноль можно считать положительным числом. Результат выполнения задания показан на рисунке 3.1.

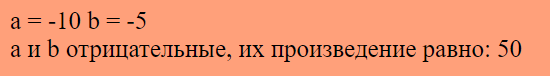
**

Рисунок 3.1 – Задание 2.1

В задании 2.2 необходимо было присвоить переменной *а* значение в промежутке [0..15]. С помощью оператора *switch* организовать вывод чисел от *a* до 15. Результат выполнения задания продемонстрирован на рисунке 3.2.



Рисунок 3.2 – Задание 2.2

В задании 2.3 необходимо было реализовать основные арифметические операции (+, -, \*, %) в виде функций с двумя параметрами.

Скрипт задания:

*<?php*

*function sum($arg1, $arg2)*

*{*

*return $arg1 + $arg2;*

*}*

*function sub($arg1, $arg2)*

*{*

*return $arg1 - $arg2;*

*}*

*function multi($arg1, $arg2)*

*{*

*return $arg1 \* $arg2;*

*}*

*function percent($arg1, $arg2)*

*{*

*return $arg1\*$arg2/100;*

*}*

*echo sub(100, 30);*

*?>*

Результат выполнения функции *sub*(100,30) продемонстрирован на рисунке 3.3.



Рисунок 3.3 – Задание 2.3

В задании 2.4 необходимо реализовать функцию с тремя параметрами: *function mathOperation*($*arg1*,$*arg*2,$*operation*), где $*arg*1, $*arg*2 – значения аргументов, $*operation* – строка с названием операции. В зависимости от переданного значения операции выполняется одна из арифметических операций из предыдущего задания и возвращается полученное значение (использовать switch).

Реализация функции:

*function mathOperation($arg1, $arg2, $operation)*

*{*

*switch ($operation) {*

*case '+':*

*return sum($arg1, $arg2);*

*case '-':*

*return sub($arg1, $arg2);*

*case '\*':*

*return multi($arg1, $arg2);*

*case '%':*

*return percent($arg1, $arg2);*

*default:*

*echo "Введена несуществующая операция";*

*break;*

*}*

*}*

Результат выполнения функции *mathOperation(10, 20, "+")* показан на рисунке 3.4.



Рисунок 3.4 – Задание 1.4

В задании 2.5 необходимо выполнить c помощью рекурсии организовать функцию возведения числа в степень. Формат: *function power*($*val*, $*pow*), где $*val* – заданное число, $*pow* – степень.

Реализация функции:

*function power($val, $pow)*

*{*

*if ($pow > 0)*

*{*

*return $val \* power($val, $pow - 1);*

*}*

*else if($pow == 0)*

*{*

*return 1;*

*}*

*else*

*{*

*return power(1 / $val, -$pow);*

*}*

*}*

Результат выполнения функции *power*(2,10) показан на рисунке 3.5.



Рисунок 3.5 – Задание 1.5

В задании 2.6 необходимо написать функцию, которая вычисляет текущее время и возвращает его в формате с правильными склонениями. Результат выполнения задания показан на рисунке 3.6.

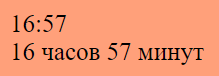


Рисунок 3.6 – Задание 1.6

# ЗАКЛЮЧЕНИЕ

*HTTP* — лёгкий в использовании расширяемый протокол. Структура клиент-сервера, вместе со способностью к простому добавлению заголовков, позволяет *HTTP* протоколу продвигаться вместе с расширяющимися возможностями Сети.

*PHP* позволяет создавать качественные *web*-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем.

*PHP* прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

Язык *PHP* постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков *web*-программирования, по крайней мере, в ближайшее время.

Мною было изучено ветвление программы, создание функций, область видимости и время жизни переменных.