ЛАБОРАТОРНАЯ РАБОТА №5
ВЗАИМОДЕЙСТВИЕ PHP И MYSQL

5.1 Язык программирования PHP

PHP – скриптовый язык программирования общего назначения, интенсивно применяемый для разработки web-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков программирования, применяющихся для создания динамических web-сайтов.

В официальной документации язык PHP подается как встраиваемый в HTML скриптовый язык с обработкой на сервере. Это позволяет сразу же иметь в виду следующее:

- обработка PHP-кода производится на стороне сервера еще до того, как web-страница будет передана браузеру; это отличает язык PHP от языка Java- Script;
- PHP-код может быть непосредственно встроен в HTML-код страницы; этим он отличается от Perl.

Кроме того, PHP – язык, который позволяет встраивать в код программы блоки HTML-кода, что позволяет использовать его для написания CGI-сценариев.

5.1.2 Переход в PHP

Механизм лексического анализа должен как-то отличать код PHP от других элементов страницы. Идентификация кода PHP называется «переходом в PHP» (escaping to PHP). Существуют четыре варианта оформления перехода в PHP:

– стандартные теги;

– короткие теги;

– теги script;

– теги в стиле ASP.

Стандартные теги используются программистами PHP чаще остальных способов, что объясняется наглядностью и удобством этой формы записи:

<?php print "Welcome to the world of PHP!": ?>

У стандартных тегов есть еще одно дополнительное преимущество: за открывающей конструкцией <? следуют символы php, однозначно определяющие тип дальнейшего кода. Это удобно при использовании в одной странице нескольких технологий, таких как JavaScript, серверные включения и PHP. Весь текст, расположенный до закрывающего тега ?>, интерпретируется как код PHP.

Короткие теги обеспечивают наиболее компактную запись для перехода в PHP:

<? print "Welcome to the world of PHP!"; ?>

По умолчанию короткие теги не используются, их нужно специально активизировать, включив параметр short_open_tag в файл php.ini. Теги script. Некоторые текстовые редакторы ошибочно принимают код PHP за код HTML, что нарушает работу над web-страницей. Проблема решается использованием тегов script:

<script language="php">

```
<?php print "Welcome to the world of PHP!"; ?>

</script>
```

Четвертый и последний способ оформления внедренного кода PHP – теги в стиле ASP (Active Server Page). Они похожи на короткие теги, описанные выше, однако вместо вопросительного знака используется знак процента (%):

```
<%php print "Welcome to the world of PHP!"; %>
```

PHP легко встраивается в HTML-код. Для обеспечения необходимой гибкости при построении динамических web-приложений можно внедрить в страницу несколько сценариев PHP. При внедрении нескольких сценариев переменные, значения которых были присвоены в одном сценарии, могут использоваться в другом сценарии той же страницы. Кроме того, код HTML может интегрироваться прямо в команды PHP (пример 5.1).

Пример 5.1

```
<HTML>

<HEAD>

  <TITLE>PHP Recipes | <? print (date("F d, Y")); ?></TITLE>

</HEAD>

  <? $big_font = "H3"; ?>

<BODY>

  <? print "<$big_font>PHP Recipes</$big_font>"; ?>

</BODY>

</HTML>
```

5.1.3 Комментарии в коде PHP

В PHP существуют два формата комментариев:

- однострочные комментарии обычно используются для коротких пояснений или примечаний, относящихся к локальному коду;
- многострочные комментарии обычно используются при оформлении алгоритмов на псевдокоде и в более подробных объяснениях.

Оба способа в конечном счете приводят к одинаковому результату и совершенно не влияют на общее быстродействие сценария. При оформлении однострочных комментариев используется два стиля комментирования. В одном случае комментарий начинается с двойного символа «косая черта» (//), а в другом – с символа фунта (#). Оба стиля работают абсолютно одинаково (пример 5.2).

Многострочные комментарии оформляются в стиле языка C – их начало и конец обозначаются символами /* и */. Многострочные комментарии особенно удобны для вывода относительно длинной сводной информации обо всем сценарии или его части (см. пример 5.2).

Пример 5.2

```
<?
  echo("Hello"); //это комментарии
  echo("Hello"); #это комментарии
  /* а это многострочные
  комментарии */
?>
```

5.1.4 Типы данных в PHP

Типы данных составляют основу любого языка программирования и являются средством, с помощью которого программист представляет разные типы информации. В PHP поддерживаются шесть основных типов данных:

- целые числа (integer);
- вещественные числа (float, double, real);
- строки (string);
- массивы (array);
- объекты (object);
- логические величины (bool).

Целое число со знаком, обычно длиной 32 бита. В PHP поддерживается запись целых чисел в десятичной (52, 14), восьмеричной (0422, 0524) и шестнадцатеричной (0x3FF, 0x22abc) форме. Вещественные числа (числа с плавающей точкой) отличаются от целых наличием дробной части. Они используются для представления значений, требующих повышенной точности, например температур или денежных величин.

В PHP поддерживаются два вещественных формата: стандартная (12.45, 98.6) и научная (3e8, 5.9736e24) запись. Строкой (string) называется последовательность символов. Строка легко может быть обработана при помощи стандартных функций, допустимо также непосредственное обращение к любому ее символу. Длина строки ограничена только размером свободной памяти, так что можно прочитать в одну строку целый файл размером около 200–300 Кбайт.

В PHP, как и в большинстве современных языков программирования, строки могут содержать служебные символы (таблица 5.1).
Строки делятся на две категории в зависимости от типа ограничителя – они могут ограничиваться парой кавычек (" ") или апострофов (' '). Между этими категориями существуют два принципиальных различия. Во-первых, имена переменных в строках, заключенных в кавычки, заменяются соответствующими значениями, а строки в апострофах интерпретируются буквально, даже если в них присутствуют имена переменных.

Таблица 5.1 – Служебные символы в PHP

Служебный символ	Назначение служебного символа
\n	Новая строка
\r	Возврат курсора
\t	Горизонтальная табуляция
\\	Обратная косая черта
\$	Знак доллара

\"	Кавычка
\[0-7]{1,3}	Восьмеричная запись числа (в виде регулярного выражения)
\x[0-9A-Fa-f]{l,2}	Шестнадцатеричная запись числа (в виде регулярного выражения)

Два следующих объявления дают одинаковый результат:

$food = "meatloaf";

$food = 'meatloaf';

Однако результаты следующих объявлений сильно различаются:

$sentence = "My favorite food is $food";

$sentence2 = 'My favorite food is $food';

Переменной $sentence присваивается строка My favorite food is meatloaf. Переменная $food автоматически интерпретируется. Переменной $sentence2 присваивается строка My favorite food is $food. В отличие от переменной $sentence, в $sentence2 осталась неинтерпретированная переменная $food. Различия обусловлены использованием кавычек и апострофов при присваивании строки переменным $sentence и $sentence2. Второе принципиальное различие заключается в том, что в строках, заключенных в кавычки, распознаются все существующие служебные символы, а в строках, заключенных в апострофы, – только служебные символы «\\» и «\».

Следующий пример наглядно демонстрирует различия между присваиванием строк, заключенных в кавычки и апострофы:

$double_list = "item1\nitem2\nitem2";

$single_list = 'item1\nitem2\nitem2';

Если вывести обе строки в браузере, окажется, что строка в кавычках содержит внутренние символы новой строки, а в строке в апострофах последовательность \n выводится как обычные символы. К отдельным символам строки можно обращаться как к элементам массива с последовательной нумерацией (пример 5.3).

Пример 5.3

$sequence_number = "04efgh";

$letter = $sequence_number[4];

В примере 5.3 переменной $letter будет присвоено значение g, т. к. нумерация элементов массивов начинается с 0. Массив представляет собой список однотипных элементов. Существует два типа массивов, различающиеся по способу идентификации элементов. В массивах первого типа элемент определяется индексом в последовательности. Массивы второго типа имеют ассоциативную природу, и для обращения к элементам используются ключи, логически связанные со значениями. Объект представляет собой переменную, экземпляр которой создается по специальному шаблону, называемому классом. Концепции объектов и классов являются неотъемлемой частью парадигмы объектно-ориентированного программирования (ООП). Логический тип данных принимает всего два значения: истинное (true) и ложное (false). Логические величины создаются двумя способами: при проверке условий и в виде значений переменных.

5.1.5 Переменные в PHP

Переменная – это область оперативной памяти, доступ к которой осуществляется по имени. Все данные, с которыми работает программа, хранятся в виде переменных.

Правила задания переменных в PHP:

- имя переменной должно начинаться со знака доллара $;
- имя переменной не должно содержать никаких других символов, кроме символов латинского алфавита, цифр и знака подчеркивания;
- имена переменных в PHP, как и в C, чувствительны к регистру символов, т. е. переменные $a и $A – это совершенно разные переменные;
- объявлять переменную можно в любом месте программы, но до места первого ее использования. При объявлении переменных тип не указывается. Выбор типа осуществляется самим интерпретатором.

Переменные в PHP могут содержать любую информацию. Исключение составляют только константы, которые могут содержать только число или строку.

5.1.5.1 Функции определения и задания типа переменных

Язык PHP предоставляет много средств для определения типа переменных. Вы можете использовать семь функций для определения типа:

– is_int($x) или is_integer($x) – возвращает true, если переданная переменная – целое число;
– is_double($x) или is_float($x) – возвращает true, если переданная переменная – вещественное число;

– is_string($x) – возвращает true, если переданная переменная – строка;

– is_array($x) – возвращает true, если переданная переменная – массив;

– is_object($x) – возвращает true, если переменная является объектом;

– is_bool($x) – возвращает true, если переменная объявлена как логическая;

– gettype(Sx) – возвращает строки, соответствующие типу переменной:

integer, double, string, object, array, bool или unknown type – если невозможно определить тип (когда тип переменной не встраивается в PHP, а добавляется с помощью модулей, расширяющих возможности языка).

Если PHP неправильно определил тип переменной, можно указать его явно. Для этого используется функция settype($x, $type), где $type – это одна из строк, возвращаемых функцией gettype() (пример 5.4). Функция settype($x, $type) возвращает значение false, если невозможно привести тип переменной $x к указанному типу $type.

Пример 5.4

```
settype($x,"double");
```

5.1.5.2 Присвоение значений. Оператор присваивания

Оператор присваивания позволяет придать переменной некоторое значение (пример 5.5).

Пример 5.5

```
$x = 4;
```

$y = $x;

$x=$x+4;

Особенности оператора присваивания:

1 Переменной можно присвоить: какое-либо значение; значение, возвращаемое функцией; значение другой переменной; значение выражения; ссылку на другую переменную.

2 В PHP нет указателей, поэтому если в переменную $x поместить файл размером 500 Кбайт, а потом присвоить ее переменной $y, то будет создана точная копия переменной $x, которая тоже будет занимать 500 Кбайт. Итого в памяти будет почти 1 Мбайт информации. Это нужно учитывать при создании так называемых временных переменных. Желательно после окончания работы с такой переменной освободить память, т. е. уничтожить переменную.

3 Интерпретатор сам выполняет преобразование типов, но иногда привести тип переменной к другому просто невозможно.

4 При присваивании создается точная копия переменной, копируется также и тип переменной. Это означает, что если массиву присвоить число, весь массив будет потерян.
5.1.5.3 Проверка существования переменной

Проверка существования переменной – это очень удобная возможность языка программирования. Благодаря возможности проверки существования переменной можно проверить, передан сценарию определенный параметр или нет, и только потом начинать с ним работать. В результате сценарий не прекратит свою работу из-за банальной ошибки пользователя. Проверка существования переменной осуществляется с помощью функции isset($x) (пример 5.6).

Пример 5.6

```
<?
 if (isset($name))
    print "Hello, $name";
 else
    print " Вы забыли ввести свое имя";
?>
```

5.1.5.4 Удаление переменных

Чтобы не засорять память, можно удалить ненужные нам переменные. Это можно делать с помощью функции unset(). Эта функция удаляет указанную в скобках переменную из памяти, и программа продолжает выполняться как будто эта переменная вообще не была инициализирована (пример 5.7).

Пример 5.7

```
<?
 $a=читаем_большой_файл;
 //обрабатываем_файл;
```

```
 unset($a); // освобождаем память
?>
```

Эта функция особенно полезна и даже необходима при обработке больших объемов данных, чтобы они не оставались в памяти компьютера и не замедляли его работу.

5.1.5.5 Логические переменные и их особенности в PHP

В PHP истиной являются: любое не равное нулю число, любая непустая строка, значение true. Значение false, пустая строка, нуль – это ложь. В использовании логических переменных в PHP имеется еще одна особенность. Если в операторах сравнения (= =, !=, <,>) один тип является логическим, то и второй также воспринимается как логический (примеры 5.8 и 5.9).

Пример 5.8

```
<?
  $x=10;
  if ($x= =1) echo "Переменная равна 1\n";
  if ($x= =true) echo "Переменная равна true\n";
?>
```

В примере 5.8 в первой строке переменной $x было присвоено значение 10. Затем $x сравнивается с 1, и если $x равно 1, то выводится строка «Переменная равна 1». Затем значение переменной $x сравнивается со значением true. И если $x равно true, то выводится строка «Переменная равна true». Исходя из приведенного выше утверждения, должна быть выведенной только вторая строка.

Пример 5.9

```
<?
  $x = 100;
  $y = true;
  echo "x= $x\n";
  echo "y= $y\n";
  if ($x==$y) echo "X=Y";
?>
```

В этом примере сначала программа сообщает, что X = 100, Y = 1, а затем, что X = Y.

5.1.6 Выражения и операции

Выражение – это последовательность знаков операций, операндов и круглых скобок, которая задает вычислительный процесс получения результата определенного типа. Операции – это специальные комбинации символов, специфицирующих действия по преобразованию различных величин. В PHP выделяют следующие виды операций:

1) арифметические;

2) строковые;

3) операции присваивания;

4) операции сравнения;

5) логические;

6) побитовые.

5.1.6.1 Арифметические операции

Как и в любом другом языке, в PHP можно использовать арифметические операции:

1) a + b – сложение;

2) a – b – вычитание;

3) a * b – умножение;

4) a / b – деление;

5) a % b – остаток от деления a на b.

Операция деления возвращает целое число (т. е. результат деления нацело), если оба выражения a и b – целого типа (или же строки, выглядящие, как целые числа), в противном случае результат будет дробным. Операция вычисления остатка от деления % работает только с целыми числами.

5.1.6.2 Строковые операции

Строковых операций в PHP всего две:

1) a.b – слияние строк a и b (пример 5.10);

2) a[n] – символ строки в позиции n.

Пример 5.10

```
$a="100";

$b="200";

echo $a.$b; //выведет 100200

echo $a + $b; //выведет 300
```

Все остальные действия над строками выполняются стандартными функциями.

5.1.6.3 Операции присваивания

В операциях присваивания правое выражение присваивается переменной слева.

1) = – присваивание;

2) +=, -=, .= и т. д. – операции сложного присваивания (пример 5.11);

3) ++, -- – инкремент и декремент. Операции операторов инкремента и декремента не работают с логическими переменными.

Пример 5.11

$message="Hello ";

$message.= "World!"; // Теперь в $message "Hello World!"

5.1.6.4 Операции сравнения

Независимо от типов своих аргументов, они всегда возвращают одно из двух значений: false или true. Операции сравнения позволяют сравнивать два значения между собой и, если условие выполнено, возвращают true, в противном случае – false. Операции сравнения (пример 5.12):

1) a == b – истина, если a равно b;

2) a != b – истина, если a не равно b;

3) a < b – истина, если a меньше b;

4) a > b – истина, если a больше b;

5) a <= b – истина, если a меньше либо равно b;

6) a >= b – истина, если a больше либо равно b;

7) = = = – истина, если a эквивалентно b;

8) ! = = – истина, если a неэквивалентно b.

Пример 5.12

$zero=0; // нуль

$tsss=""; // пустая строка
if ($zero==$tsss) echo "Переменные равны";

if ($zero===$tsss) echo "Переменные эквивалентны";

/* в данном примере на экран будет выведена только первая строка, что переменные равны */

В PHP 5 сравнивать на равенство или неравенство можно не только скалярные переменные (т. е. строки и числа), но также массивы и объекты. При сравнении массива сравнивается отдельно каждая пара элементов.

5.1.6.5 Логические операции

Эти операции предназначены исключительно для работы с логическими выражениями и также возвращают false или true:

1) ! a – истина, если a ложно, и наоборот;

2) a && b – истина, если истинны и a, и b;

3) a | | b – истина, если истинны или a, или b, или оба операнда.

Вычисление логических выражений, содержащих такие операции, идет всегда слева направо, при этом если результат уже очевиден, то вычисления обрываются, даже если в выражении присутствуют вызовы функции.

5.1.6.6 Приоритет операций

Приоритет арифметических и логических операций представлен в таблице 5.2. Операции с более высоким приоритетом выполняются в первую очередь. Изменить приоритет операции можно с помощью круглых скобок.

Таблица 4.2 – Приоритет арифметических и логических операций

Приоритет	Оператор	Порядок выполнения
13	(постфикс)++ (постфикс) --	слева направо
12	(префикс)++ (префикс) --	справа налево
11	* / %	слева направо
10	+ -	
9	<< >>	
8	< <= > >=	
7	= = !=	
6	&	
5	^	
4	|	
3	&&	
2	||	
1	= += -= *= /= %= >>= <<= &= ^= |=	справа налево

5.1.7 Конструкции PHP (операторы выбора, операторы цикла)

5.1.7.1 Условный оператор (if … else)

Формат условного оператора:
if (логическое_выражение)

инструкция_1;

else

инструкция_2;

Если логическое_выражение истинно, то выполняется инструкция_1, а иначе – инструкция_2. Как и в любом другом языке, конструкция else может опускаться. Если инструкция_1 или инструкция_2 должны состоять из нескольких команд, то они, как всегда, заключаются в фигурные скобки. В условном операторе инструкция_1 и инструкция_2, в свою очередь, могут быть условными, что позволяет организовать цепочки проверок любой степени вложенности. Синтаксис языка предполагает, что при вложенных условных операторах каждое else соответствует ближайшему if. Существует альтернативная форма конструкции if … else:

if (логическое_выражение):

инструкция_1;

elseif (другое_логическое_выражение):

инструкция_2;

else:

инструкция_3;

endif

Обратите внимание на расположение двоеточия. Если его пропустить, то будет сгенерировано сообщение об ошибке. Блоки else и elseif можно опускать.

Пример 5.13

В форму вводятся два числа, и скрипт выводит введенные числа, определяет наибольшее из них и выводит их среднее арифметическое.

```
<HTML>
<HEAD>
  <TITLE> Определение наибольшего числа </TITLE>
</HEAD>
<BODY>
<?
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
  echo "Число 1: ".$_POST["chislo1"]."<BR>";
  echo "Число 2: ".$_POST["chislo2"]."<BR>";
  if ($_POST["chislo1"]>$_POST["chislo2"])
   echo "Наибольшее число ".$_POST["chislo1"]."<BR>";
  elseif ($_POST["chislo1"]<$_POST["chislo2"])
   echo "Наибольшее число ".$_POST["chislo2"]."<BR>";
  else
   echo "Числа равны! <BR>";
  $sr=($_POST["chislo1"]+$_POST["chislo2"])/2;
  echo "Среднее арифметическое ".$sr."<BR>";
}
else {
?>
<H1> Форма для ввода чисел </H1>
<FORM METHOD= "POST" ACTION= "form.php" >
<TABLE>
<TR><TD>Введите первое число:</TD>
    <TD><INPUT NAME ="chislo1" TYPE="TEXT" > </TD>
</TR>
```

```
<TR><TD>Введите второе число:</TD>
    <TD><INPUT NAME ="chislo2" TYPE="TEXT" > </TD>
</TR>
<TR>
  <TD COLSPAN=2 >
    <INPUT TYPE="SUBMIT" VALUE="Отправить">
  </TD>
 </TR>
</TABLE>
</FORM>
<? }?>
</BODY>
</HTML>
```

5.1.7.2 Переключатель (switch)

Переключатель switch является более удобным средством для организации множественного выбора, чем оператор if … else.

Синтаксис переключателя:

```
switch (expression)  // переключающее выражение
{
 case value1: // константное выражение 1
  statements1; // блок операторов
  break;
 case value2: // константное выражение 2
  statements2;
  break;
 default:
  statements;
}
```

Управляющая структура switch передает управление тому из помеченных операторов case, для которого значение константного выражения совпадает со значением переключающего выражения. Сначала анализируется переключающее выражение expression и осуществляется переход к той ветви программы, для которой его значение совпадает с константным выражением. Далее следует выполнение оператора или группы операторов до тех пор, пока не встретится ключевое слово break, которым обозначается выход из

переключателя. Если же значение переключающего выражения не совпадает ни с одним из константных выражений, то выполняется переход к оператору, помеченному меткой default.

В каждом переключателе может быть не более одной метки default. Ключевые слова break и default не являются обязательными и их можно опускать.

Пример 5.14

В форме пользователю, к примеру, нужно ввести ответ на вопрос «Продолжить работу программы?». Если пользователь наберет в строке ввода «yes», то работа будет продолжена, если «no», то завершена.

```
<?
switch($answer)
{  case "yes": echo ("Продолжаем работу! ");
  // далее следуют операторы, которые будут выполняться в этом случае
     break;
   case "no": echo ("Завершаем работу");
     break;
   default: echo ("Некорректный ввод");
     break;
} ?>
```

Существует альтернативная форма конструкции switch:

```
switch (expression):
  case value1: statements1; break;
  case value2: statements2; break;
  default: statements;
endswitch;
```

5.1.7.3 Циклы

В PHP определены 4 разных оператора цикла:

- цикл с предусловием (while);
- цикл с постусловием (do … while);
- итерационный цикл (for);
- итерационный цикл (foreach).

Цикл с предусловием while

Синтаксис цикла с предусловием:

```
while (логическое_выражение)
```

инструкция;

Принцип работы цикла с предусловием:

- вычисляется значение логического выражения;
- если значение истинно, выполняется тело цикла, в противном случае переходим на следующий за циклом оператор.

Альтернативный синтаксис цикла с предусловием:

while (логическое_выражение):

инструкция;

endwhile;

Цикл с предусловием do … while

Синтаксис цикла с постусловием:

do {

команды;

} while (логическое_выражение);

Цикл с постусловием отличается от цикла с предусловием тем, что сначала выполняется тело цикла, а только потом проверяется условие. Таким образом, тело цикла хотя бы один раз, но будет обязательно выполнено. Альтернативного синтаксиса для цикла do … while нет.

Итерационный цикл for Итерационный цикл (или цикл со счетчиком) используется для выполнения тела цикла определенное количество раз. Синтаксис итерационного цикла

for:

for (инициал-щие_команды; условие_цикла; команды_после_прохода)

тело_цикла;

Оператор for начинает свою работу с выполнения команд инициализации. Данные команды выполняются всего лишь один раз. После этого проверяется условие: если оно истинно, выполняется тело цикла. После того как будет выполнен последний оператор тела, выполняются «команды после перехода». Затем снова проверяется условие, в случае, если оно истинно, выполняется тело цикла и поститерационные команды и т. д. Альтернативный синтаксис итерационного цикла:

for (инициал-щие_команды; условие_цикла; команды_после_прохода):

тело_цикла;

endfor;

Итерационный цикл foreach

Синтаксис итерационного цикла foreach:

foreach (массив as $ключ=>$значение)

тело_цикла;

Здесь команды циклически выполняются для каждого элемента массива, при этом очередная пара ключ=>значение оказывается в переменных $ключ и $значение.

Пример 5.15

Вывод всех переменных окружения
<? foreach ($_SERVER as $k=>$v)

```
   echo "<b>$k</b> => <tt>$v</tt><br>\n";
```

?>

У цикла foreach имеется и другая форма записи, которую следует применять, когда не интересует значение ключа очередного элемента. В этом случае доступно лишь значение очередного элемента массива, но не его ключ. Выглядит она следующим образом:

```
foreach ($массив as $значение)
   тело_цикла;
```

5.2 Подключение к MYSQL базе данных

SQL может применяться в прикладных программах двумя способами: в виде встроенного SQL и интерфейса программирования приложений (Application Program Interface, API). Первый способ напоминает использование PHP – инструкции SQL размещаются среди кода прикладной программы. В настоящий момент такой стиль не поддерживают ни MySQL, ни PHP. Второй подход заключается в том, что программа взаимодействует с СУБД посредством совокупности функций. Именно такой подход используется при взаимодействии PHP и MySQL.

5.2.1 Функции управления соединением с сервером MySQL

Для установки соединения с сервером MySQL используется функция mysql_connect(). Синтаксис функции:

```
resource mysql_connect ([string server [, string username [, string password]]])
```

Эта функция устанавливает соединение с сервером server MySQL и возвращает дескриптор соединения с базой данных, по которому все другие функции, принимающие этот дескриптор в качестве аргумента, будут однозначно определять выбранную базу данных. Вторым и третьим аргументами этой функции являются имя пользователя username и его пароль password соответственно (пример 5.16).

Пример 5.16

```
<?php
$dblocation = "localhost"; // Имя сервера
$dbuser = "root";     // Имя пользователя
$dbpasswd = "";      // Пароль
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx) // Если дескриптор равен 0, соединение не установлено
{
```

```
 echo("<P>В настоящий момент сервер базы данных не доступен, поэтому
     корректное отображение страницы невозможно.</P>");
 exit();
}?>
```

В примере 5.16 переменные $dblocation, $dbuser и $dbpasswd хранят имя сервера, имя пользователя и пароль и, как правило, прописываются в отдельном файле (к примеру, config.php), который потом вставляется в каждый PHP-файл, имеющий код для работы с MySQL. Для закрытия открытого ранее соединения можно воспользоваться функцией mysql_close(). Открытое соединение закрывается автоматически по завершении работы сценария. Синтаксис функции:

```
bool mysql_close ([resource conn_id])
```

Эта функция разрывает соединение с сервером MySQL и возвращает true при успешном выполнении операции и false в противном случае. Функция принимает в качестве аргумента дескриптор соединения с базой данных, возвращаемый функцией mysql_connect() (пример 5.17).

Пример 5.17

```
<? // устанавливаем соединение с базой данных
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
 // Выводим предупреждение
 echo ("<P>В настоящий момент сервер базы данных не доступен, поэтому
 корректное отображение страницы невозможно.</P>");
 // Завершаем работу в случае неудачи
 exit();
}
 // Выполняем запросы
if(mysql_close($dbcnx)) // разрываем соединение
{
 echo("Соединение с базой данных прекращено");
}
else
{
 echo("Не удалось завершить соединение");
} ?>
```

5.2.2 Функции состояния и диагностики ошибок

Функции mysql_errno() и mysql_error() возвращают числовой код и информационное сообщение об ошибке в работе связанных с MySQL функций PHP. Однако эти функции не могут использоваться для получения информации о результатах безуспешного завершения вызовов функций mysql_connect(), поскольку идентификатор связи становится доступным только после успешного установления соединения.

Синтаксис функции:
int mysql_errno ([resource conn_id);

Функция возвращает код ошибки для связанной с заданным соединением MySQL функции, которая последней возвращала значение состояния. Нулевой код указывает на отсутствие ошибки.

Синтаксис функции:

string mysql_error ([resource conn_id]) ;

Функция возвращает строку с сообщением об ошибке для связанной с заданным соединением MySQL функции, которая последней возвращала значение состояния. Пустая строка свидетельствует об отсутствии ошибки.

5.2.3 Функции построения и выполнения запросов

Функции построения и выполнения запросов используются для создания и отсылки запросов на сервер MySQL. Строки запроса должны состоять из одного оператора SQL и не должны заканчиваться символом «точка с запятой» (";") или последовательностью "\g". Окончания «;» и «\g» являются соглашениями клиентской программы MySQL и не используются при выводе запросов через PHP.

Выборка указанной базы данных, для того чтобы сделать ее текущей базой данных для заданного соединения, выполняется функцией mysql_select_db(). Синтаксис функции:

bool mysql_select_db(string db_name [, resource conn_id]);

Таблицы, на которые будут делаться ссылки в последующих запросах, по умолчанию будут принадлежать этой базе данных, если не будет указана другая база данных. Функция возвращает значение «true» при успешном завершении и «false», если появилась ошибка.

Пример 5.18

Переменные $dblocation, $dbuser и $dbpasswd хранят имя сервера, имя пользователя и пароль и, как правило, прописываются в отдельном файле (к примеру, config.php) вместе с функциями для соединения и выбора базы данных, который потом вставляется в каждый PHP-файл, где имеется код для работы с MySQL.

```
<?php

$dblocation = "localhost";

$dbname = "softtime";

$dbuser = "root";

$dbpasswd = "";
```

```
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
 echo( "<P>В настоящий момент сервер базы данных не доступен, поэтому
     корректное отображение страницы невозможно.</P>" );
 exit();
}
if (!@mysql_select_db($dbname, $dbcnx))
{
 echo( "<P>В настоящий момент база данных не доступна, поэтому
     корректное отображение страницы невозможно.</P>" );
 exit(); } ?>
```

Для отправки строки к серверу MySQL для заданного соединения используется функция mysql_query(). Синтаксис функции:

int mysql_query(string query [, resource conn_id]) ;

Для операторов DELETE, INSERT, REPLACE и UPDATE функция mysql_query () возвращает значение «true» в случае успешного выполнения и значение «false», если имела место ошибка. После успешного выполнения запроса можно вызвать функцию mysql_affected_rows () , чтобы узнать число измененных строк. Для операторов SELECT функция mysql_query () возвращает положительный идентификатор результирующего набора в случае успешного выполнения и значение «false», если имела место ошибка. Возвращаемый после успешного выполнения запроса идентификатор результата может использоваться самыми разными функциями обработки результирующих наборов, считывающими аргумент result_id. Для освобождения занятых результирующим набором ресурсов системы этот идентификатор необходимо передать функции

mysql_free_result ().

5.2.4 Функции обработки результирующих наборов

Описанные в данном разделе функции используются для выборки результатов выполнения запросов, а также для предоставления информации о результатах, например количества измененных строк.

Синтаксис функции:

array mysql_fetch_array (resource result_id [, int result_type]) ;

Функция возвращает следующую строку данного результирующего набора в виде массива. Возвращает значение «false», если строк больше не осталось. Возвращаемый массив содержит значения, идентифицируемые как по числовым индексам столбцов, так и по ключам имен столбцов. Другими словами, доступ к значению каждого столбца можно получить как по числовому индексу, так и по названию столбца. Учитывается регистр

символов при написании ассоциативных индексов, поэтому их следует записывать в том же регистре, что и имена столбцов в запросе.

Предположим, например, что отправлен следующий запрос:

SELECT last_name, first_name FROM president

Если строки извлекаются из результирующего набора в массив с именем $row, доступ к его элементам можно получить следующим образом:

$row [0] или $row ["last_name"] Содержит значение столбца last_name.
$row [1] или $row ["first_name"] Содержит значение столбца first_name.

Параметр result_type может иметь значения MYSQL_ASSOC (возвращаются значения только по индексам имен), MYSQL_NUM (возвращаются значения только по числовым индексам) или MYSQL_BOTH (возвращаются значения по индексам обоих типов). Если этот аргумент отсутствует, его значение по умолчанию устанавливается равным MYSQL_BOTH.

Вызов mysql_fetch_array () с параметром result_type, имеющим значение MYSQL_ASSOC или MYSQL_NUM, эквивалентно вызову mysql_fetch_assoc () или mysql_f etch_row ().

Пример 5.19

Из таблицы authors базы данных forums выбираются все записи, содержащие информацию об авторах, и выводятся на экран в табличной форме. Результат работы скрипта показан на рисунке 5.1.

```
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
 // Определяем таблицу и заголовок
 echo "<table border=1>";
 echo "<tr><td>имя</td><td>пароль</td><td>e-mail</td><td>url</td></tr>";
 // Так как запрос возвращает несколько строк, применяем цикл
 while($author = mysql_fetch_array($ath))
 {
  echo "<tr><td>".$author['name']." </td><td>".$author['passw']."
    </td><td>".$author['email']." </td><td>".
  $author['url']."] </td></tr>";
 }
 echo "</table>";
```

```
}
else
{
 echo "<p><b>Error: ".mysql_error()."</b><p>";
 exit();
}
                                                    ?>
```

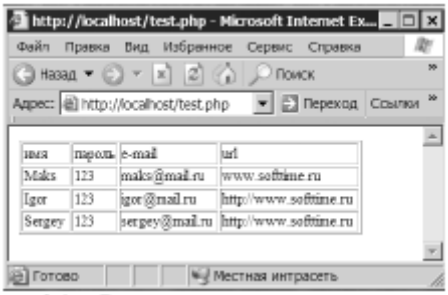

Рисунок 6.1 – Результат выполнения примера 5.19

Пример 5.20. Создать форму для поиска книг в базе данных bookby (poisk.php). На странице должно быть поле со списком, в котором пользователь может выбрать критерий поиска (по ISBN, или по автору, или по заголовку книги), текстовое поле для ввода искомого значения и кнопка отправки данных. Например:

```
<HTML> <HEAD> <TITLE>Поиск книги</TITLE> </HEAD>
<BODY>
<H1>Поиск книги в каталоге</H1>
<FORM ACTION="results.php" METHOD="POST">
 Выберите параметр поиска:<BR>
 <SELECT NAME="searchtype">
  <OPTION VALUE="author">Автор
  < OPTION VALUE ="title">Заголовок
  < OPTION VALUE ="isbn">ISBN
 </SELECT>
 <BR>
 Введите параметры поиска:<BR>
 <INPUT NAME="searchterm" TYPE="text">
 <BR>
 <INPUT TYPE="submit" VALUE="Поиск">
</FORM>
```

```
</BODY>
</HTML>
```

Пример 5.21. Выводит на экран результаты поиска results.php по критериям, введенным в форму, созданную в задании 6.2. После нажатия кнопки в форме в переменной searchtype будет храниться имя столбца, по которому будет осуществляться поиск в таблице books, а искомое значение будет храниться в переменной searchterm. Пример файла results.php:

```
<HTML>
 <HEAD> <TITLE>Результаты поиска книги</TITLE> </HEAD>
<BODY>
<H1> Результаты поиска книги </H1>
<?
trim($_POST["searchterm"]) ; //убираем лишние пробелы по краям слова
if (!$_POST["searchtype"] || !$_POST["searchterm"])
{
  echo "Вы не ввели критерии поиска. Вернитесь назад и попробуйте еще раз";
  exit();
}
// добавляет символы косой черты перед символами,
// которые могут быть интерпретированы как управляющие
$_POST["searchtype"] = addslashes($_POST["searchtype"]);
$_POST["searchterm"] = addslashes($_POST["searchterm"]);
include "config.php"; // соединяемся с сервером MySQL и выбираем БД
// составляем запрос
$query = "select * from books where ".$_POST["searchtype"]." like
'%".$_POST["searchterm"]."%'";
$result = mysql_query($query); // выполняем запрос
// определяем количество строк в запросе
$num_results = mysql_num_rows ($result);
echo "<P>Количество найденных книг: ".$num_results."</P>";
for ($i=0; $i <$num_results; $i++) {
 $row = mysql_fetch_array($result); // функция берет каждую строку
//из списка результата и возвращает ее в виде ассоциативного массива
 echo "<P><STRONG>". ($i+1) .". Название: ";
```

```
 echo $row["title"];
 echo "</STRONG><BR>Автор: " ;
 echo $row["author"];
 echo "<BR>ISBN: ";
 echo $row["isbn"];
 echo "<BR>Цена: ";
 echo $row["price"];
 echo "</P>";
}
?>
</BODY> </HTML>
```

Функция mysql_fetch_assoc () возвращает следующую строку данного результирующего набора в виде ассоциативного массива. Синтаксис функции:

array mysql_fetch_assoc (resource result_id);

Функция возвращает значение «false», если строк больше не осталось. Значения столбцов доступны с помощью ассоциативных имен, соответствующих именам столбцов, выбранных запросом. Вызов mysql_fetch_assoc () подобен вызову mysql_fetch_array() со вторым аргументом MYSQL_ASSOC. Значения столбцов с цифровыми индексам не возвращаются. Функция mysql_fetch_row() возвращает следующую строку заданного результирующего набора в виде массива либо значение «false», если строк больше не осталось. Синтаксис функции:

array mysql_fetch_row (resource result_id) ;

Доступ к значениям столбцов можно получать через элементы массива, используя индексы столбцов в диапазоне от 0 до mysql_num_fields ()-1. Вызов функции mysql_fetch_row() аналогичен вызову функции mysql_fetch_array() со вторым аргументом, равным MYSQL_NUM. Значения столбцов с ассоциативными индексами не возвращаются. Функция mysql_affected_rows () возвращает количество строк, измененных последним запросом DELETE, INSERT, REPLACE или UPDATE в заданном соединении. Синтаксис функции:

int mysql_affected_rows ([resource conn_id]) ;

Функция mysql_affected_rows () возвращает значение 0, если ни одна строка изменена не была, и значение -1, если имела место ошибка. Вызванная после оператора SELECT функция mysql_affected_rows () возвращает число строк, полученных в результате выборки. Хотя обычно для операторов SELECT используется функция mysql_num_rows (). Функция mysql_num_fields () возвращает число столбцов в заданном результирующем наборе. Синтаксис функции:

int mysql_num_fields (resource result_id) ;

Функция mysql_num_rows () возвращает число строк в заданном результирующем наборе. Синтаксис функции:

int mysql_num_rows (resource result_id) ;

Функция mysql_insert_id () возвращает значение AUTO_INCREMENT, сгенерированное последним запросом заданного соединения. Синтаксис функции:

int mysql_insert_id ([resource conn_id]) ;

Если в процессе всего соединения такое значение создано не было, эта функция возвращает нуль. В общем, функцию mysql_insert_id () следует вызывать сразу после запроса, который сгенерирует значение AUTO_INCREMENT.

Если между этим запросом и вызовом функции будет выполнен промежуточный запрос, возвращаемое функцией значение может быть сброшено до нуля.

Задание

Написать скрипты, которые выполняют нижеперечисленные запросы и выводят на экран результаты их работы.

Вариант 1

1. Выбрать все сведения о сотруднике из таблицы Сотрудники и отсортировать результат по Коду сотрудника.

2.Выбрать из таблицы Сотрудник Должность и Подразделение, а из таблицы Дети выбрать имя Ребенка соответствующего Сотрудника.

3. Выбрать из таблицы Дети Имена, Даты рождения и Код ребенка Сотрудников, значения Фамилии которых начинаются с ‘Иванов’.

4. Вывести список Стоимостей подарков и Даты выдачи из таблицы Подарки, у которых стоимость укладывается в диапазон 200 – 300 рублей.

5. Вывести список Сотрудников из таблицы Сотрудники, у которых Фамилия начинаются на букву ‘К’.

6. Вывести список Сотрудников из таблицы Сотрудники, у которых названия Подразделения начинаются со слова ‘Отдел’ и Должность – ‘Заместитель’.

7. Вывести суммарную стоимость Подарков в выбранную Дату выдачи подарков.

8. Вывести среднюю цену подарков и общее их количество для Сотрудника с Фамилией ‘Иванов’.

9. Вывести общую сумму Подарков и поместить результат в поле с названием Sum_gift, полученные Ребенком с Именем ‘Магдалина’.

10. Вывести список Сотрудников, Дети которых носят имена ‘Марк’, ‘Антоний’, ‘Аврелий’.

Вариант 2

1. Выбрать из таблицы Приборы все поля и отсортировать результат по Названию прибора (по возрастанию) и Тип прибора (по убыванию).

2. Выбрать из таблицы Приборы названия приборов и дату производства, а из таблицы Учет приема выбрать имя соответствующего мастера.

3. Вывести список названий прибора из таблицы Учет приема мастеров, которые разряд не выше 4.

4. Вывести список ФИО владельца прибора из таблицы Мастеров, у которых фамилия начинается на одну из букв диапазона ‘В’ – ‘Г’.

5. Вывести ФИО владельца прибора из таблицы Учет приема у, которых содержат в Виде поломки содержится слово ‘гарант’.

6. Вывести список названий приборов и Фамилии мастеров у, которых Дата приема в ремонт в период с 01.01.2018 по 31.01.2018.

7. Вывести общую стоимость ремонтов для владельца за период с 01.01.2018 по 31.01.2018.

8. Вывести все сведения о приборе (все поля таблицы Приборы), а также ФИО мастера с минимальной общей стоимостью ремонта.

9. Вывести список приборов (все поля из таблицы Приборы) за последний месяц.

10. Вывести список Названий приборов, у которых были Стоимость ремонта по цене 150 руб.

Вариант 3

1. Выбрать из таблицы Цветы список его характеристик Название, Сорт и Средняя высота и отсортировать результат по полю Средняя высота.

2. Выбрать из таблицы Цветы Названия и Тип листа, а из таблицы Продажи выбрать Код продавца и Код продажи.

3. Вывести список Названий из таблицы Цветы, которые были проданы в день, кроме издательства ’08.03.2017’.

4. Вывести список названий цветов и Цену продажи, которые были проданы с кодами 3, 7, 9, 11.

5. Выбрать Фамилии продавца из таблицы Продавцы, у которых значение оканчивается на ‘ский’.

6. Вывести список Названий цветов, которых были проданы продавцом ‘Иванова’.

7. Вывести количество лет с момента приема на работу и имя соответствующего продавца.

8. Вывести все сведения о Цветке, а также максимальной общей стоимостью.

9. Вывести список продавцов, дата приема, которых меньше заданного пользователем.

10. Вывести список Названий цветов и Цену продажи в каждой из цветов и поместить результат в курсор с названием Temp1.

Вариант 4

1. Выбрать все поля из таблицы Лекарственные средства таким образом, чтобы в результате порядок столбцов был следующим: Название лекарства, Единицы измерения, Количество в упаковке и все остальные в исходном порядке.

2. Выбрать из справочника поставщиков названия компаний, телефоны и Юридический адрес, у которых полное название компании начинается с ‘ОАО’.

3. Вывести все данные из таблицы Продажи, у которых дата поставки находится в диапазоне 01.01.2018 – 01.02.2018.

4. Вывести список лекарств, которые выпущены следующими поставщиками: ‘БелФарма’, ‘ОАО Фармин’, ‘ЧТУП Биоветпром’.

5. Выбрать коды поставщиков, даты поставки и код поступления, если количество в поставке больше 100 или цена за единицу находится в диапазоне от 20 до 50.

6. Вывести список поставщиков, которые поставили лекарства за период ’01.01.2018’ по ’01.02.2018’.

7. Вывести общую сумму поставок лекарств, выполненных ‘ЗАО Оптторг’.

8. Вывести название лекарств, суммарную стоимость партии, поместив в результат в поле с названием Itogo, в поставках за период с 01.01.2018 по 01.02.2018.

9. Вывести список названий компаний-поставщиков и поместить результат в курсор с названием Temp2.

10. Вывести список поставщиков, у которых были приобретена партия в количестве более 1000.

Вариант 5

1. Выбрать все поля из таблицы Списания таким образом, чтобы в результате порядок столбцов был следующим: Код сотрудника, Причина списания, Дата списания и Код оборудования.

2. Выбрать из таблицы Оборудование Название оборудования и его тип, а из таблицы Списания Дату списания, у которых название оборудования начинается со слова ‘Одноразовый’.

3. Вывести список названий оборудования и место установки, которые были закуплены в период с 12.03.2017 по 15.06.2017.

4. Вывести список названий оборудования, которые списаны сотрудниками с фамилией: ‘Петров’, ‘Иванов’, ‘Сидоров’.

5. Выбрать код оборудования, имена сотрудников, название оборудования, если код списания находится в диапазоне от 10 до 25 и место установки не “Цех”.

6. Вывести список оборудования и причины списания, которые были списаны ‘Ивановым’.

7. Вывести общее количество всего списанного оборудования, списанных в период с 01.01.2018 по 01.02.2018.

8. Вывести количество лет с момента поступления на работу и Фамилию сотрудника.

9. Вывести список Фамилий сотрудников и названия оборудования, которые они списали.

10. Вывести список оборудования и поместить результат в курсор с названием Temp3.

Вариант 6

1. Выбрать все поля из таблицы Блюдо таким образом, чтобы в результате порядок столбцов был следующим: Порядок приготовления, Количество углеводов, Количество калорий и все остальные в исходном порядке.

2. Выбрать из справочника продукты названия и Ед. измерения, у которых название начинается с ‘Консервированный’.

3. Вывести все данные из таблицы Блюда, у которых Объем продукта любого использованного в приготовлении в пределах 0,2-0,3 литра.

4. Вывести список блюд, у которых в рецепте используются продукты: ‘Филе ягненка’, ‘Лимон’, ‘Чеснок’.

5. Выбрать блюда, если количество углеводов больше 100 или Количество калорий находится в диапазоне от 300 до 400.

6. Название продукта и его объем, у которого Ед измерения ’шт.’.

7. Вывести общий объем продуктов, если вес блюда более ‘400’.

8. Вывести Тип блюда и суммарный объем продуктов, поместив в результат в поле с названием Itogo, если Ед измерения “гр.”.

9. Вывести список блюд и поместить результат в курсор с названием Temp3.

10. Вывести список продуктов, у которых суммарный объем в более 500.

Вариант 7

1. Выбрать все поля из таблицы Регистраторы таким образом, чтобы в результате порядок столбцов был следующим: Должность, Дата приема и все остальные в исходном порядке.

2. Выбрать из справочника организации полное названия компании, телефон и Юридический адрес, у которых полное название компании начинается с ‘ОАО’.

3. Вывести все данные из таблицы Регистрация документов, у которых дата регистрации находится в диапазоне 01.01.2018 – 01.02.2018.

4. Вывести список Регистраторов, которые организация отправитель: ‘БелФарма’, ‘БНТУ’, ‘БелЧип’.

5. Выбрать коды регистраторов, даты приема на работу и полное название организации, если Тип документа “Письмо” или Дата регистрации в периоде с 01.01.2018 по 01.02.2018.

6. Вывести список организаций, которые отправили документы за период ’01.01.2018’ по ’01.02.2018’.

7. Вывести общую количество отправленных документов, отправленных ‘БНТУ’.

8. Вывести Фамилию Регистратора, общее количество отправленных документов, поместив в результат в поле с названием Itogo, за период с 01.01.2018 по 01.02.2018.

9. Вывести список названий организаций отправителей и поместить результат в курсор с названием Temp4.

10. Вывести список организаций, у которых есть номер белорусских операторов.

Вариант 8

1. Выбрать все поля из таблицы Сотрудники таким образом, чтобы в результате порядок столбцов был следующим: Должность, Дата приема и все остальные в исходном порядке.

2. Выбрать из справочника Статьи названия Статьи, Причина увольнения и Номер статьи увольнения, у которых Причина увольнения содержит слово ‘халатность’.

3. Вывести все данные из таблицы Сотрудники, у которых дата приема на работу находится в диапазоне 01.01.2018 – 01.02.2018.

4. Вывести список Сотрудников, у которых Причина увольнения: ‘По собственному желанию’, ‘Халатность на рабочем месте’, ‘Окончание действия контракта’.

5. Выбрать Фамилия, даты приема на работу и Причина увольнения, если Номер статьи 181 или Денежная компенсация от 100 до 200 руб.

6. Вывести список сотрудников, которые уволенных за период ’01.01.2018’ по ’01.02.2018’.

7. Вывести общую количество уволенных по статье 181.

8. Вывести Номер статьи увольнения, общее количество сотрудников, уволенных по этой статье, поместив в результат в поле с названием Itogo, за период с 01.01.2018 по 01.02.2018.

9. Вывести список причин увольнений и поместить результат в курсор с названием Temp4.

10. Вывести список сотрудников, у которых была Денежная компенсация.

Вариант 9

1. Выбрать все сведения о сотруднике из таблицы Сотрудники и отсортировать результат по Дате приема на работу.

2. Выбрать из справочника Сотрудники названия Фамилию, Должность и Подразделение, у которых отпуск длился более 20 дней.

3. Вывести все данные из таблицы Сотрудники, у которых есть льготы по отпуску.

4. Вывести список Сотрудников, у которых Должность содержит слова: ‘Главный’, ‘Руководитель’, ‘Заместитель’.

5. Выбрать Фамилия и дату приема на работу, если Дата окончания отпуска большей ’01.02.2018’ или Оплата отпуск от 500 до 600 руб.

6. Вывести список сотрудников, которые ходили в отпуск за период ’01.01.2018’ по ’01.02.2018’.

7. Вывести общую сумму оплат за отпуск у сотрудников с Фамилией “Иванов”.

8. Вывести Фамилию, общее количество дней отпусков, поместив в результат в поле с названием Itogo, за период с 01.01.2018 по 01.02.2018.

9. Вывести список Типы отпусков и поместить результат в курсор с названием Temp4.

10. Вывести список сотрудников, у которых была Денежная компенсация.

Вариант 10

1. Выбрать все поля из таблицы Регистраторы таким образом, чтобы в результате порядок столбцов был следующим: Должность, Дата приема и все остальные в исходном порядке.

2. Выбрать из справочника организации полное названия компании, телефон и Юридический адрес, у которых полное название компании начинается с ‘ОАО’.

3. Вывести все данные из таблицы Регистрация документов, у которых дата регистрации находится в диапазоне 01.01.2018 – 01.02.2018.

4. Вывести список Регистраторов, которые организация отправитель: ‘БелФарма’, ‘БНТУ’, ‘БелЧип’.

5. Выбрать коды регистраторов, даты приема на работу и полное название организации, если Тип документа “Письмо” или Дата регистрации в периоде с 01.01.2018 по 01.02.2018.

6. Вывести список организаций, которые отправили документы за период ’01.01.2018’ по ’01.02.2018’.

7. Вывести общую количество отправленных документов, отправленных ‘БНТУ’.

8. Вывести Фамилию Регистратора, общее количество отправленных документов, поместив в результат в поле с названием Itogo, за период с 01.01.2018 по 01.02.2018.

9. Вывести список названий организаций отправителей и поместить результат в курсор с названием Temp4.

10. Вывести список организаций, у которых есть номер белорусских операторов.

Вариант 11

1. Выбрать все сведения о сотруднике из таблицы Сотрудники и отсортировать результат по Дате приема на работу.

2. Выбрать из справочника Сотрудники названия Фамилию, Должность и Подразделение, у которых разница между Дата приема на работу более 1 года.

3. Вывести все данные из таблицы Сотрудники, у которых есть льготы на должности.

4. Вывести список Сотрудников, у которых Должность содержит слова: ‘Главный’, ‘Руководитель’, ‘Заместитель’.

5. Выбрать Фамилия и дату приема на работу, если возраст приема на работу 30 лет или есть требования к квалификации.

6. Вывести список сотрудников, которые назначили на должность за период ’01.01.2018’ по ’01.02.2018’.

7. Вывести количество назначений у сотрудников с Фамилией “Иванов”.

8. Вывести Фамилию, общее количество назначений с льготами, поместив в результат в поле с названием Itogo, за период с 01.01.2018 по 01.02.2018.

9. Вывести список Названия должностей и поместить результат в курсор с названием Temp4.

10. Вывести список сотрудников, у которых есть Требования к квалификации.

Вариант 12

1. Выбрать все поля из таблицы Клиент таким образом, чтобы в результате порядок столбцов был следующим: Серия и номер паспорта, Код клиента, Телефон и все остальные в исходном порядке.

2. Выбрать из таблицы Оборудование Название оборудования и его тип, а из таблицы Прокат Дату начала проката, у которых название оборудования начинается с букв ‘Лыж’.

3. Вывести список названий оборудования и его типа, у которых Дата поступления в период с 12.03.2017 по 15.06.2017.

4. Вывести список названий оборудования, которые брались клиентом с фамилией: ‘Петров’, ‘Иванов’, ‘Сидоров’.

5. Выбрать код оборудования, имена клиентов, название оборудования, если стоимость проката находится в диапазоне от 10 до 25 и Тип оборудования не “Холодильник”.

6. Вывести список оборудования и Даты начала проката, которые были взяты в прокат ‘Ивановым’.

7. Вывести общую стоимость, потраченную на прокат оборудования в период с 01.01.2018 по 01.02.2018.

8. Вывести количество лет с момента поступления в прокат оборудования и его тип.

9. Вывести список Фамилий клиентов и названия оборудования, которые они брали.

10. Вывести список оборудования и поместить результат в курсор с названием Temp3.

Вариант 13

1. Выбрать все поля из таблицы Списания таким образом, чтобы в результате порядок столбцов был следующим: Дата поступления в прокат, Название оборудования, Тип оборудования и Код оборудования.

2. Выбрать из таблицы Оборудование Название оборудования и его тип, а из таблицы Списания Дату списания, у которых название оборудования начинается со слова ‘Межконтинентальный’.

3. Вывести список названий оборудования и Фамилию сотрудника, которые списал оборудования в период с 12.03.2017 по 15.06.2017.

4. Вывести список названий оборудования, которые списаны сотрудниками с фамилией: ‘Петров’, ‘Иванов’, ‘Сидоров’.

5. Выбрать код оборудования, имена сотрудников, название оборудования, а дата поступления в прокат в диапазоне с 01.01.2018 по 01.02.2018 и Должность сотрудника начинается со слов “Заместитель”.

6. Вывести список оборудования и причины списания, которые были списаны ‘Ивановым’.

7. Вывести общее количество всего списанного оборудования, списанных в период с 01.01.2018 по 01.02.2018.

8. Вывести количество лет с момента поступления на работу и Фамилию сотрудников.

9. Вывести список Фамилий сотрудников и названия оборудования, которые они списали.

10. Вывести список оборудования и поместить результат в курсор с названием Temp3.

Вариант 14

1. Выбрать из таблицы Цветы список его характеристик Название, Сорт и Средняя высота и отсортировать результат по полю Названию.

2. Выбрать из таблицы Цветы Названия и Тип листа, а из таблицы Продажи выбрать Дату поступления и Цену за единицу.

3. Вывести список Названий из таблицы Цветы, которые были проданы в день, кроме издательства ’08.03.2017’.

4. Вывести список названий цветов и Цену продажи (Цена за единицу х Количество).

5. Выбрать Полное названия из таблицы Поставщики, у которых Сокращенное название начинается на ‘ОАО’.

6. Вывести список Названий цветов, которых были приобретены у ‘Марифлора’.

7. Общую сумму продаж по каждому из цветов.

8. Вывести все сведения о Цветке, а также максимальную продажу.

9. Вывести список цветов, средняя высота, у которого меньше заданного пользователем.

10. Вывести список Полное название организации и ФИО руководителя и поместить результат в курсор с названием Temp1.

Вариант 15

1. Выбрать все поля из таблицы Клиент таким образом, чтобы в результате порядок столбцов был следующим: Серия и номер паспорта, Код клиента, Телефон и все остальные в исходном порядке.

2. Выбрать из таблицы Номера Цену за сутки и его тип, а из таблицы Бронирование Дату заезда, у которых в перечне удобств есть ‘Джакузи’.

3. Вывести список клиентов, которые заезжают в период с 01.01.2018 по 01.02.2018.

4. Вывести список личных данных о клиентах, у которых фамилия: ‘Петров’, ‘Иванов’, ‘Сидоров’.

5. Выбрать тип номера, фамилия клиента, а дата выезда в диапазоне с 01.01.2018 по 01.02.2018 и Стоимость их проживания более 1000.

6. Вывести список типов номеров, которые снимал ‘Иванов’.

7. Вывести общее заработок в период с 01.01.2018 по 01.02.2018.

8. Вывести количество лет с момента последнего выезда клиента и его ФИО.

9. Вывести список Фамилий сотрудников и номер с максимальной ценой за сутки, которые они снимали.

10. Вывести список ФИО клиентов и поместить результат в курсор с названием Temp3.

Вариант 16

1. Выбрать все поля из таблицы Клиент таким образом, чтобы в результате порядок столбцов был следующим: Серия и номер паспорта, Код клиента, Телефон и все остальные в исходном порядке.

2. Выбрать из таблицы Оборудование Название оборудования и его тип, а из таблицы Прокат Дату возврата, у которых название оборудования начинается с букв ‘Конь’.

3. Вывести список названий оборудования и Фамилий клиентов, у которых Дата возврата в период с 12.03.2017 по 15.06.2017.

4. Вывести список названий оборудования, которые брались клиенты с фамилией: ‘Петров’, ‘Иванов’, ‘Сидоров’.

5. Выбрать код оборудования, имена клиентов, название оборудования, если штраф находится в диапазоне от 10 до 25 и Тип оборудования не “Холодильник”.

6. Вывести список оборудования и Даты возврата, которые были взяты в прокат ‘Ивановым’.

7. Вывести общую стоимость штрафов, уплаченных в прокат оборудования в период с 01.01.2018 по 01.02.2018.

8. Вывести количество лет с момента поступления в прокат оборудования и общее количество возвратов.

9. Вывести список Фамилий клиентов и названия оборудования, которые они возвращали.

10. Вывести список оборудования и поместить результат в курсор с названием Temp3.

Вариант 17

1. Выбрать все сведения о сотруднике из таблицы Сотрудники и отсортировать результат по Дате приема на работу.

2.Выбрать из таблицы Сотрудник Должность и Фамилию, а из таблицы Ценности выбрать Название ценности, за которую он несет ответственность.

3. Выбрать из таблицы Учет ценностей, Дату Постановки на учет и Должность Сотрудников, значения Фамилии которых начинаются с ‘Иванов’.

4. Вывести список Закупочных стоимостей и Сроков гарантии из таблицы Ценности, у которых стоимость укладывается в диапазон 200 – 300 рублей и данные сгруппировать по Фамилии сотрудника.

5. Вывести список Сотрудников из таблицы Сотрудники, у которых Фамилия начинаются на букву ‘К’.

6. Вывести список Сотрудников из таблицы Сотрудники, у которых названия Должности начинаются со слова ‘Заместитель’ и Дата приема на работу ранее 01.08.2017.

7. Вывести суммарную стоимость ценностей за которую несет ответственности Сотрудники.

8. Вывести общее количество ценностей за которую несет ответственность Сотрудник с Фамилией ‘Иванов’.

9. Вывести общую сумму Подарков и поместить результат в поле с названием Itogo, за которую несет ответственность сотрудник с Именем ‘Марк’ и должностью, которое содержит слово ‘Начальник’.

10. Вывести список Сотрудников, которые несут ответственность за ценности, в названии которых есть слова ‘ПК’, ‘Грифельный’, ‘Расточной’.

Вариант 18

1. Выбрать все сведения о сотруднике из таблицы Сотрудники и отсортировать результат по Должности.

2.Выбрать из таблицы Сотрудник Должность и Подразделение, а из таблицы Ремонт выбрать Название ремонтной работы и ее стоимость.

3. Выбрать из таблицы Сотрудники Фамилию и Разряд мастера, а из таблицы Ремонт Количество дней ремонта, значения Фамилии которых начинаются с ‘Иванов’.

4. Вывести список Стоимостей ремонта и Количество дней ремонта из таблицы Ремонт, у которых стоимость укладывается в диапазон 20 – 50 рублей.

5. Вывести список Стоимости ремонта и Название и стоимость каждого из этапов данного ремонта.

6. Вывести список Фамилий из таблицы Мастеров, у которые выполняли ремонт с названием содержащем слова ‘Плановый’ и Разряд мастера выше 4.

7. Вывести суммарную стоимость Ремонтов за выбранную Дату приема.

8. Вывести среднюю цену ремонтов, выполненных мастером за месяц и общее их количество для Мастера с Фамилией ‘Иванов’.

9. Вывести общую сумму Этапов работ и поместить результат в поле с названием Itogo, выполненные мастером с Именем ‘Антонио’.

10. Вывести список Сотрудников, которые выполняли этапы работ, содержащие в названии слова ‘Капитальный’, ‘Плановый’.

Вариант 19

1. Выбрать все поля из таблицы Лекарственные средства и отсортировать результат по Количеству в упаковке.

2. Выбрать из справочника Лекарственные средства данные у которых Название производителя начинается с ‘ОАО’.

3. Вывести Лекарственные средства, у которых дата продажи находится в диапазоне 01.01.2018 – 01.02.2018.

4. Вывести список лекарств, которые выпущены следующими производителями: ‘БелФарма’, ‘ОАО Фармин’, ‘НПК Биотест ООО’.

5. Выбрать Лекарственные средства, если количество в упаковке 100 или цена за единицу находится в диапазоне от 20 до 50.

6. Вывести ФИО кассиров, которые продали лекарства за период ’01.01.2018’ по ’01.02.2018’ с Названием лекарства ‘Но ШПа’.

7. Вывести общую сумму проданных лекарств, производителя ‘БелФарма’.

8. Вывести название лекарств, суммарную стоимость продаж, поместив в результат в поле с названием Itogo, в поставках за период с 01.01.2018 по 01.02.2018.

9. Вывести список названий компаний-производителей и поместить результат в курсор с названием Temp2.

10. Вывести список производителей, у которых лекарственные общее количество проданных единиц более 5.