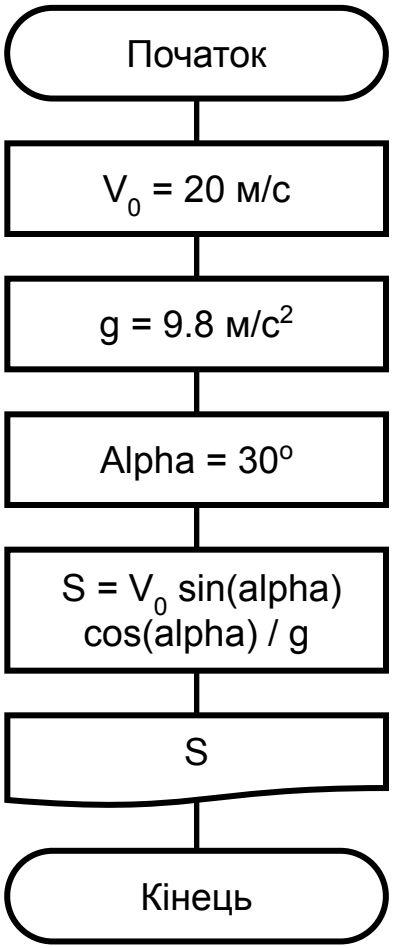
КУРС **PHP**

УПРАВЛІННЯ ХОДОМ ПРОГРАМИ

ЛІНІЙНИЙ




```
<?php
$v0 = 20;
$g = 9.8;
$alpha = 30;
$s = v0 * sin(alpha) * cos(alpha) / g;
echo “s = $s”;
?>
```

РОЗГАЛУЖЕНИЙ

```
if (вираз) блок_виконання;

if (вираз) блок_виконання1;
else блок_виконання2;

if (вираз) блок_виконання1;
elseif(вираз) блок_виконання2;
...
else блок_виконанняN;
```

РОЗГАЛУЖЕННЯ

```
<? $a=5;

$b=3;

if($a==5 && $b!=0 ):

$c = 100 + $a / $b;

echo "$c";

endif;

?>
```

Цей приклад може бути записаний у стандартному синтаксисі C:

```
<?

if($a==5 && $b!=0) {

$c = 100 + $a / $b;

echo $c;

}

?>
```

ОПЕРАТОРИ ПОРІВНЯННЯ

Позначення	Назва	Опис	Приклад
==	Рівність	Значення змінних рівні	$a == $b
===	Еквівалентність	Рівні значення і типи змінних	$a === $b
!=	Нерівність	Значення змінних нерівні	$a != $b
<>	Нерівність		$a <> $b
!==	Нееквівалентність	Змінні не еквівалентны	$a !== $b
<	Менше		$a < $b
>	Більше		$a > $b
<=	Менше або дорівнює		$a <= $b
>=	Більше або дорівнює		$a >= $b

ЛОГІЧНІ ОПЕРАТОРИ

Позначення	Назва	Опис	Приклад
and	І (AND)	$a і $b істинні (True)	$a and $b
&&	І (AND)		$a && $b
or	АБО (ЧИ) (OR)	Хочаб одна із змінних $a або $b істинна (можливо обидві)	$a or $b
||	АБО (ЧИ) (OR)		$a || $b
xor	Виключне або (XOR)	Одна із змінних істинна. Проте не обидві	$a xor $b
!	Інверсія (NOT)	Якщо $a=True, то !$a=False і навпаки	! $a

ОПЕРАТОР SWITCH

```
switch (вираз чи змінна){
case значення1: блок_дій1; break;
case значення2: блок_дій2; break;
  ...
default: блок_дій_по_замовчанню;
}
```

ЦИКЛИ

```
while (вираз) {блок_виконання}

do {блок_виконання} while (вираз);

for (вираз1; вираз2; вираз3)
{блок_виконання}
```

WHILE (ВИРАЗ) {БЛОК_ВИКОНАННЯ}

Значення виразу перевіряється до виконання ітерації:

```
<? $a=0;

$list[$a]=$a;

while($a <100) {

$a++;

$list[$a]=$a;

echo "$list[$a]  ";

}

?>
```

Результат

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

DO {БЛОК_ВИКОНАННЯ} WHILE (ВИРАЗ);

1. $i = 0;

do {

print $i;

} while ($i>0); //(доки)

Значення виразу перевіряється після ітерації. Цей цикл хоча б один раз виконується.

результат : 0

ЦИКЛИ FOR - НАЙБІЛЬШ МОГУТНІЙ ЦИКЛ У PHP.

Вони працюють подібно їх аналогам у C. Синтаксис циклу FOR :

FOR (expr1; expr2; expr3) statement

перший вираз (expr1) безумовно , обчислюється (виконується) на початку циклу .

На початку кожної ітерації обчислюється expr2. Якщо воно дорівнює TRUE, то цикл продовжується й виконуються вкладені оператори . Якщо воно дорівнює FALSE, то цикл закінчується . Якщо він порожній, то цикл продовжується нескінченно.

Наприкінці кожної ітерації обчислюється expr3.

ПРИКЛАДИ

```
<?/* приклад 1 */
echo" приклад 1 ";
for ($i = 1; $i <= 10; $i++) {
print $i;
}
/* приклад 2 */
echo" <br> приклад 2 ";
for ($i = 1;;$i++) {
if ($i > 10) { break;
}
print $i;
}
```

```
/* приклад 3 */

echo" <br> приклад 3 ";
$i = 1;
for (;;) {
if ($i > 10) {
break;
}
print $i;
$i++;
}
/* приклад 4 */
echo"<br> приклад 4 ";
for ($i = 1; $i <= 10; print $i, $i++) ; ?>
```

Результат

приклад 1 12345678910

PHP також підтримує альтернативний синтаксис FOR :

FOR (expr1; expr2; expr3): вирази ; ...; endfor;.

BREAK ПЕРЕРИВАЄ ВИКОНАННЯ ПОТОЧНОГО ЦИКЛУ .

```
<? $i = 0;

while ($i < 10) {

if ($i == 5) {

print $i ;

$i = 7;

print $i ;

break;

}

print $i ;

$i++;

} ?>
```

результат : 0123457

CONTINUE ПЕРЕХОДИТЬ НА ПОЧАТОК НАЙБЛИЖЧОГО ЦИКЛУ

```
<? $i = 0;
while ($i < 10)
{if ($i == 5){
print $i;
$i=7;
print $i;
continue;
}
print $i;
$i++; }
?>
```

результат 0123457789

КОНСТРУКЦІЯ SWITCH

```
<? $a=0;
while ($a < 5) {
switch($a) {
case 1;
echo "a is 1<br>";
break;
case 2;
echo "a is 2<br>";
break;
default;
echo "a is unknown
<br>";
break; }
echo "$a <br>";
$a++; }?>
```

Результат

a is unknown	0
a is 1	1
a is 2	2
a is unknown	3
a is unknown	4

REQUIRE

require “імя файла”;

Інструкція дозволяє включити файл **до** виконання сценарію.

Інтерпретатор, знайшовши *require*, просто заміняє його вмістом файла.

Зручно використовувати цю інструкцію для задання

Файл header.html:

```
<html><head>
<title>My Company's Official Web Page</title>
</head>
<body>
```

Сценарий:

```
require "header.html"
/* тело документа */
```

INCLUDE

Оператор INCLUDE вставляє й виконує вміст зазначеного файлу під час виконання сценарію. Це відбувається щораз , коли зустрічається оператор INCLUDE, так що ви можете включити цей оператор всередину циклу , щоб включити

```
for ($i=0; $i<4; $i++) {
include "header{$i}.html";
}
```

Якщо у Вас є 4-ри файла html, кожен в залежно від назви файла буде мати цифру від 0 до 3, то результат: 0123.

Завжди цей оператор поміщайте у фігурні дужки, інакше програма буде працювати не коректно.

Якщо маємо декілька операторів, то повторюватися 4-ри рази буде перший оператор, а потім після завершення циклу 2-й і 3-й.

```
for ($i=0; $i<4; $i++)
operator1;
operator2;
operator3;
```

ІНСТРУКЦІЇ ОДНОРАЗОВОГО ВКЛЮЧЕННЯ

INCLUDE_ONCE **REQUIRE_ONCE**

Використовуються для попередження повторного включення файлів, що приводить до помилок.

Перед включенням файла інтерпретарор перевіряє,чи був уже включеним цей файл, якщо так, то файл не буде включеним у склад сценарія.