Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы трансляции

ОТЧЕТ
к лабораторной работе №1
на тему

ОПРЕДЕЛЕНИЕ МОДЕЛИ ЯЗЫКА. ВЫБОР ИНСТРУМЕНТАЛЬНОЙ ЯЗЫКОВОЙ СРЕДЫ

БГУИР 1-40 04 01

Студент П. Н. Носкович

Преподаватель Н. Ю. Гриценко

Минск 2025

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Цель данной работы заключается в формировании подмножества языка программирования, включая все типы констант, переменных, операторов и функций. В рамках этого подмножества должны быть представлены все доступные типы данных, а также учтены все структуры, доступные в рассматриваемом языке. Также необходимо рассмотреть возможности подключения библиотек (модулей) и других ресурсов.

В работе будут проанализированы операторы циклов, такие как *do...while* и *for*, которые предоставляет язык, а также условные операторы, включая *if...else* и *case*.

Кроме того, будет определена инструментальная языковая среда, включающая язык программирования с указанием версии, операционную систему (например, *Windows* или *Linux*), на которой будет происходить разработка, а также тип компьютера (*PC* или *Macintosh*).

В отчете по лабораторной работе будет представлено полное описание подмножества языка программирования, а также тексты 3 программ, которые включают все элементы этого подмножества. Также будет приведено детальное описание инструментальной языковой среды.

2 ПОДМНОЖЕСТВО ЯЗЫКА ПРОГРАММИРОВАНИЯ

В языке программирования *PHP* существует множество элементов, которые позволяют разрабатывать различные типы программ. В этом разделе мы подробно рассмотрим подмножество языка, включающее типы данных, переменные, операторы циклов, условные операторы и структуры данных, с конкретными примерами.

2.1 Типы данных в языке программирования *PHP*

PHP является языком с динамической типизацией. Это значит, что тип данных переменной выводится во время выполнения, и в отличии от ряда других языков программирования в *PHP* не надо указывать перед переменной тип данных. В *PHP* есть несколько базовых типов данных. [1]

1 Скалярный тип *int* представляет целое число со знаком. Переменная типа *int* занимает в памяти 32 бита, то есть может принимать значения от -2147483648 до 2147483647. Если переменная получает числовое значение вне этого диапазона, то она трактуется как переменная типа *float*.

2 Тип *float* представляет числа с плавающей точкой. Размер числа с плавающей точкой зависит от платформы. Максимально возможное значение, как правило, составляет 1.8E+308 с точностью около 14 десятичных цифр.

3 Тип *string* представляет строки. Для работы с текстом можно применять строки. Строки бывают двух типов: в двойных кавычках и одинарных. От типа кавычек зависит обработка строк интерпретатором. Так, переменные в двойных кавычках заменяются значениями, а переменные в одинарных кавычках остаются неизменными.

4 Логический тип *bool* представляет два значения: *true* (истина) или *false* (ложь). Используется для логических операций.

5 Любой тип *mixed* представляет значение, которое может быть любого типа. Этот тип часто используется в сигнатурах функций, когда возможен любой тип данных.

2.2 Операторы циклов и условные операторы

Рассмотрим условные операторы языка.

1 Оператор *if*. Оператор *if* используется для выполнения блока кода, если заданное условие истинно. Синтаксис представлен на рисунке 1.

```php
$a = 5;
if($a>0){
    echo "Переменная а больше нуля";
}
elseif($a < 0){
    echo "Переменная а меньше нуля";
}
else{
    echo "Переменная а равна нулю";
}
```

Рисунок 1 – Синтаксис оператора *if*

2 Оператор *switch..case*. Оператор *switch..case* позволяет выбрать один из нескольких блоков кода на основе значения выражения. Синтаксис представлен на рисунке 2.

```php
$a = 3;
switch($a)
{
    case 1:
        echo "сложение";
        break;
    case 2:
        echo "вычитание";
        break;
    default:
        echo "действие по умолчанию";
        break;
}
```

Рисунок 2 – Синтаксис оператора *switch..case*

3 В языке программирования *PHP* существуют операторы сравнения, такие как == (равно), != (не равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), которые позволяют сравнивать значения различных типов данных. Эти операторы возвращают логическое значение (*true* или *false*) в зависимости от результата сравнения.

4 Операторы логического выражения. В *PHP* доступны логические операторы *and* (логическое И), *or* (логическое ИЛИ), ! (логическое НЕ). Они используются для комбинирования и инвертирования логических значений.

В языке программирования *PHP* доступны следующие операторы цикла:

1 Оператор цикла *for*. Оператор *for* в *PHP* используется для повторения блока кода определенное количество раз или до выполнения заданного

условия. Синтаксис оператора можно увидеть на рисунке 3.

```php
<?php
for ($i = 1; $i < 10; $i++)
{
    echo "Квадрат числа $i равен " . $i * $i . "<br/>";
}
?>
```

Рисунок 3 – Синтаксис оператора *for*

2 Оператор цикла *do...while*. используется для создания цикла, который повторяет выполнение блока кода, пока заданное условие истинно. Синтаксис можно увидеть на рисунке 4.

```php
<?php
$counter = 1;
do
{
    echo $counter * $counter . "<br />";
    $counter++;
}
while($counter<10)
?>
```

Рисунок 4 – Синтаксис оператора *do...while*

3 Операторы управления циклом. Оператор *break* используется для немедленного выхода из цикла. Оператор *continue* используется для пропуска текущей итерации цикла и перехода к следующей итерации. Пример использования операторов *continue* и *break* можно увидеть на рисунке 5.

```php
<?php
for ($i = 1; $i < 10; $i++)
{
    if($i==5)
    {
        continue;
    }
    echo "Квадрат числа $i равен " . $i * $i . "<br/>";
}
?>
```

Рисунок 5 – Синтаксис операторов управления циклами

2.3 Структуры

В *PHP* реализованы пользовательские структуры. Структуры (или

пользовательские типы данных) в *PHP* создаются с помощью классов и они позволяют объединять различные типы данных в единую единицу. Это особенно полезно, когда необходимо хранить связанные данные, которые могут иметь разные типы. Пользовательские структуры позволяют разработчикам создавать более сложные и организованные модели данных. Пример объявления структуры представлен на рисунке 6.

```php
<?php
class Person
{ }

$person = new Person();
print_r($person);
?>
```

Рисунок 6 – Пример структуры в языке *PHP*

Массивы в *PHP* – это структуры данных, которые позволяют хранить наборы значений одного или разных типов в виде единой переменной. Они удобны для работы с большими объемами данных и упрощают код, так как позволяют обращаться к элементам массива по индексам или ключам. В *PHP* массивы могут быть индексированными (с числовыми индексами) или ассоциативными (с именованными ключами). Пример использования массивов представлен на рисунке 7.

```php
<?php
$numbers = [1, 4, 9, 16];
echo $numbers[2];   // 9
?>
```

Рисунок 7 – Пример массива в языке *PHP*

В *PHP* существуют функции, которые позволяют организовывать код, делая его более структурированным и переиспользуемым. Функции в *PHP* могут принимать аргументы, выполнять вычисления и возвращать результаты. Они поддерживают как позиционные аргументы, так и аргументы с указанием имени. Также функции могут иметь значения по умолчанию для аргументов.

3 ИНСТРУМЕНТАЛЬНАЯ ЯЗЫКОВАЯ СРЕДА

В качестве языковой среды выбран язык программирования *Fortran* (версия 90/95). Разработка основана на работе с операционной системой *Windows* на ПК.

Инструментальная среда *Fortran* представляет собой интегрированную среду разработки, которая предоставляет программистам возможности для создания, отладки и тестирования программ на языке *Fortran*. Рассмотрим возможности языковой среды.

Инструментальная среда *Fortran* включает в себя редакторы кода, такие как *Code::Blocks* или *Visual Studio Code*, которые предлагают различные функции, включая подсветку синтаксиса, автодополнение кода, инструменты отладки и рефакторинга.

Для разработки программ на *Fortran* необходим компилятор, который может преобразовывать исходный код на *Fortran* в исполняемый файл. Популярные компиляторы, такие как *GNU Fortran (gfortran)*, позволяют создавать исполняемые файлы для различных платформ и архитектур, что упрощает кросс-компиляцию.

Инструментальная среда *Fortran* также включает в себя средства для тестирования, которые позволяют разработчикам писать и запускать тесты для проверки функциональности кода. Это особенно важно для обеспечения качества и надежности программного обеспечения.

Кроме того, *Fortran* имеет поддержку управления зависимостями, что упрощает процесс работы с внешними библиотеками и пакетами.

В данной главе была рассмотрена основная языковая среда для языка *Fortran*, а также выбрана платформа, на которой будет проводиться разработка и анализ.

ЗАКЛЮЧЕНИЕ

В ходе работы было определено подмножество языка программирования *PHP*, включая типы, константы, переменные, операторы и функции. В подмножество языка были включены все основные типы данных, такие как *int, float, string* и *bool*, операторы цикла и условные операторы. Учтены структурные типы, включая пользовательские структуры, что позволяет группировать связанные данные. Определена инструментальная языковая среда, т.е. язык программирования и операционная система для разработки. В ходе лабораторной работы дается полное определение подмножества языка программирования, тексты трёх программ, включающих все элементы этого подмножества. Приводится подробное описание инструментальной языковой среды.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Руководство по *PHP* [Электронный ресурс]. – Режим доступа: https://metanit.com/php/tutorial/.

ПРИЛОЖЕНИЕ А
(обязательное)
Текст программ

Листинг 1 – module1.php

```
<?php
define("PI", 3.14159);

$integerVar = 42;
$floatVar = 3.14;
$stringVar = "Hello, PHP!";
$boolVar = true;
$arrayVar = [1, 2, 3, 4, 5];
$nullVar = null;

  echo "PI: " . PI . "\n";
  echo "Integer: " . $integerVar .
"\n";
  echo "Float: " . $floatVar . "\n";
  echo "String: " . $stringVar . "\n";
  echo "Boolean: " . ($boolVar ?
'true' : 'false') . "\n";
  echo "Array: " . implode(", ",
$arrayVar) . "\n";
  echo "Null: " . (is_null($nullVar) ?
'null' : 'not null') . "\n";

  $sum = $integerVar + $floatVar;
  $product = $integerVar * $floatVar;
  echo "Sum: " . $sum . "\n";
  echo "Product: " . $product . "\n";

  if ($integerVar > 0) {
      echo "Integer is positive.\n";
  } else {
      echo "Integer is non-
positive.\n";
  }

  switch ($integerVar) {
      case 42:
          echo "The answer to life,
the universe, and everything.\n";
          break;
      default:
          echo "Just a number.\n";
  }
?>
```

Листинг 2 – module2.php

```
<?php
  function factorial($n) {
      if ($n <= 1) {
          return 1;
      } else {
          return $n * factorial($n -
1);
      }
  }

  echo "Factorials using for loop:\n";
```

```
  for ($i = 1; $i <= 5; $i++) {
      echo "Factorial of $i is " .
factorial($i) . "\n";
  }

  echo "Factorials using do...while
loop:\n";
  $j = 1;
  do {
      echo "Factorial of $j is " .
factorial($j) . "\n";
      $j++;
  } while ($j <= 5);

  $numbers = [1, 2, 3, 4, 5];
  $squaredNumbers =
array_map(function($n) {
      return $n * $n;
  }, $numbers);

  echo "Squared numbers: " .
implode(", ", $squaredNumbers) . "\n";
?>
```

Листинг 3 – main.php

```
<?php
  require 'vendor/autoload.php';

  $data = ['name' => 'PHP', 'version' => '8.0'];
  $jsonString = json_encode($data);
  echo "JSON: " . $jsonString . "\n";

  $filename = "example.txt";

  file_put_contents($filename, "Hello, PHP File Handling!\n");

  $fileContent = file_get_contents($filename);
  echo "File Content: " . $fileContent;

  unlink($filename);
?>
```