

Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Усынин Максим Валерьевич
Должность: Ректор
Дата подписания: 04.06.2022 16:49:58
Уникальный программный ключ:
f498e59e83f65dd7c3ce7bb8a25cbbab033ebc36

**Частное образовательное учреждение высшего образования
«Международный Институт Дизайна и Сервиса»
(ЧОУВО МИДиС)**

Кафедра математики и информатики

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ДИСЦИПЛИНЕ

ОП.05 ОСНОВЫ ПРОГРАММИРОВАНИЯ

Специальность:

09.02.03 Программирование в компьютерных системах

Уровень образования обучающихся:

Основное общее образование

Вид подготовки:

Базовый

Челябинск 2022

Методические рекомендации по дисциплине ОП.04 Основы алгоритмизации и программирования разработаны на основе федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование, утверждено приказом Министерства образования и науки РФ от 9 декабря 2016 года № 1547 и рабочей программы дисциплины ОП.04 Основы алгоритмизации и программирования.

Автор-составитель: Прилепина Е.В.

Методические рекомендации по дисциплине ОП.04 Основы алгоритмизации и программирования рассмотрены и одобрены на заседании кафедры педагогики и психологии, протокол № 10 от 30.05.2022 г.

Заведующий кафедрой математики и информатики

Л.Ю. Овсяницкая

СОДЕРЖАНИЕ

1.Пояснительная записка.....	4
2.Методические рекомендации по темам.....	5
3.Рекомендуемая литература.....	13

1. Пояснительная записка

Методические рекомендации составлены в соответствии с рабочей программой дисциплины ОП.04 Основы алгоритмизации и программирования и предназначены для реализации основной профессиональной образовательной программы среднего профессионального образования (далее – образовательной программы) по специальности 09.02.07 Информационные системы и программирование.

Целями методических указаний по дисциплине ОП.04 Основы алгоритмизации и программирования являются:

- систематизация и закрепление полученных теоретических знаний;
- углубление и расширение теоретических знаний;
- развитие познавательных способностей и активности студентов, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации.

В методических рекомендациях изложены рекомендации к темам, приведены примерные задания, перечень литературы, рекомендуемой для выполнения заданий.

В результате освоения дисциплины ОП.04 Основы алгоритмизации и программирования обучающийся должен сформировать:

Общие компетенции

Код	Наименование общих компетенций
ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 02	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.
ОК 04	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.
ОК 09	Использовать информационные технологии в профессиональной деятельности.
ОК 10	Пользоваться профессиональной документацией на государственном и иностранном языках

В результате изучения дисциплины ОП.04 Основы алгоритмизации и программирования обучающиеся должны:

уметь:

- Разрабатывать алгоритмы для конкретных задач.
- Использовать программы для графического отображения алгоритмов.
- Определять сложность работы алгоритмов.
- Работать в среде программирования.
- Реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.
- Оформлять код программы в соответствии со стандартом кодирования.
- Выполнять проверку, отладку кода программы.

знать:

- Понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции.
- Эволюцию языков программирования, их классификацию, понятие системы программирования.

- Основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти.
- Подпрограммы, составление библиотек подпрограмм
- Объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения

Формы и методы контроля работы студентов: выборочная/фронтальная проверка выполненных заданий; устный опрос, тестирование, заслушивание сообщений (докладов), проверка презентаций, проведение экзамена.

Критерии оценки результатов работы студентов:

- «отлично», если работа выполнена в полном объёме с соблюдением необходимой последовательности. Обучающиеся работают полностью самостоятельно: подбирают необходимые для выполнения предлагаемых работ источники знаний, показывают необходимые для проведения практической работы теоретические знания, практические умения и навыки.
- «хорошо», если работа выполняется обучающимися в полном объёме и самостоятельно. Допускаются отклонения от необходимой последовательности выполнения, не влияющие на правильность конечного результата. Работа показывает знание обучающимися основного теоретического материала и овладение умениями, необходимыми для самостоятельного выполнения работы. Могут быть неточности и небрежность в оформлении результатов работы.
- «удовлетворительно», если работа выполняется и оформляется обучающимися при сторонней помощи. На выполнение работы затрачивается много времени (можно дать возможность доделать работу дома). Обучающиеся показывают знания теоретического материала, но испытывают затруднение при самостоятельной работе.
- «неудовлетворительно» выставляется в том случае, когда обучающиеся не подготовлены к выполнению работы. Полученные результаты не позволяют сделать правильных выводов и полностью расходятся с поставленной целью. Показывается плохое знание теоретического материала и отсутствие необходимых умений.

2. Методические рекомендации по темам

Тема 1.2. Типы данных

Задание. Используя список рекомендуемой литературы, подробно изучите тему лекционного материала и самостоятельно решите задачи в тетради.

1. Определите значение переменной *a* после исполнения данного алгоритма:

$a := 36$

$b := a / 12$

$b := b + a / 4$

$a := a / b * 3$

Порядок действий соответствует правилам арифметики.

В ответе укажите одно число – значение переменной *a*.

2. Определите значение переменной *b* после исполнения данного алгоритма:

$a := 81$

$b := a / 3$

$a := b + 2 * a$

$b := a / 9 * 3$

Порядок действий соответствует правилам арифметики.

В ответе укажите одно число – значение переменной *b*.

3. Исполнитель Май4 преобразует число, записанное на экране. У исполнителя две команды, которым присвоены номера:

1. Прибавь 1
2. умножь на 2

Первая из них увеличивает число на экране на 1, вторая увеличивает это в 2 раза. Программа для исполнителя Май4 – это последовательность команд.

Сколько есть программ, которые число 11 преобразуют в число 30?

Вычислить:

4. Алгоритм вычисления значения функции $F(n)$, где n – натуральное число, задан следующими соотношениями:

$$F(1) = 1$$

$$F(n) = F(n-1) * (n + 1), \text{ при } n > 1$$

Чему равно значение функции $F(5)$? В ответе запишите только целое число.

5. У исполнителя Квадр две команды, которым присвоены номера:

1. прибавь 1,
2. возведи в квадрат.

Первая из этих команд увеличивает число на экране на 1, вторая – возводит в квадрат. Программа для исполнителя Квадр - это последовательность номеров команд.

Запишите программу для исполнителя Квадр, которая преобразует число 5 в число 2500 и содержит не более 6 команд. Если таких программ более одной, то запишите любую из них.

6. Исполнитель КУЗНЕЧИК живёт на числовой оси. Начальное положение КУЗНЕЧИКА – точка 0. Система команд Кузнечика:

Вперед 4 – Кузнечик прыгает вперед на 4 единицы,

Назад 3 – Кузнечик прыгает назад на 3 единицы.

Какое наименьшее количество раз должна встретиться в программе команда «Назад 3», чтобы Кузнечик оказался в точке 27?

7. Исполнитель Робот действует на клетчатой доске, между соседними клетками которой могут стоять стены. Робот передвигается по клеткам доски и может выполнять команды 1 (вверх), 2 (вниз), 3 (вправо) и 4 (влево), переходя на соседнюю клетку в направлении, указанном в скобках. Если в этом направлении между клетками стоит стена, то Робот разрушается. Робот успешно выполнил программу

3233241

Какую последовательность из трех команд должен выполнить Робот, чтобы вернуться в ту клетку, где он был перед началом выполнения программы, и не разрушиться вне зависимости от того, какие стены стоят на поле?

8. Автомат получает на вход четырёхзначное число. По этому числу строится новое число по следующим правилам.

1. Складываются первая и вторая, а также третья и четвёртая цифры исходного числа.
2. Полученные два числа записываются друг за другом в порядке убывания (без разделителей).

Пример. Исходное число: 3165. Суммы: $3 + 1 = 4$; $6 + 5 = 11$. Результат: 114.

Укажите наименьшее число, в результате обработки которого, автомат выдаст число 1311.

9. У исполнителя Аккорд две команды, которым присвоены номера:

1. отними 1
2. умножь на x

где x – неизвестное положительное число. Выполняя первую из них, Аккорд отнимает от числа на экране 1, а выполняя вторую, умножает это число на x .

Программа для исполнителя Аккорд – это последовательность номеров команд.

Известно, что программа 12121 переводит число 4 в число 23. Определите значение x .

Тема 2.1. Операторы языка программирования

1. Решение задач «Простейшие алгоритмы»

1. Какое значение примет переменная С в результате выполнения фрагмента алгоритма

A:=5; B:=10; C:=7	1) 0
если A>=B	2) 2
то C:=A*B+3	3) 7
иначе C:=B/A+7	4) 53
все	5) 46
C:=C-7	

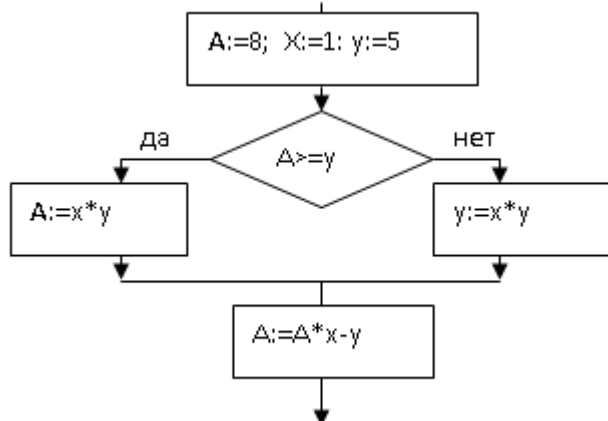
2. Придя из школы, Петя обычно бросает монетку и, в зависимости от того, что выпадет - орел или решка, - идет либо в кино, либо в парк. Действует Петя по следующему алгоритму:
БРОСИТЬ МОНЕТУ

Если ОРЕЛ то ИДТИ В КИНО иначе ИДТИ В ПАРК все
УЧИТЬ УРОКИ

Однажды, монетка закатилась в щель и встала на ребро. Что будет делать Петя?

- 1) УЧИТЬ УРОКИ
- 2) ИДТИ В КИНО
- 3) ИДТИ В ПАРК
- 4) 4) ИДТИ В КИНО, затем УЧИТЬ УРОКИ
- 5) ИДТИ В ПАРК, затем УЧИТЬ УРОКИ

3. Выполни алгоритм и выпиши значение переменной А:



4. Цепочки символов (строки) создаются по следующему правилу.

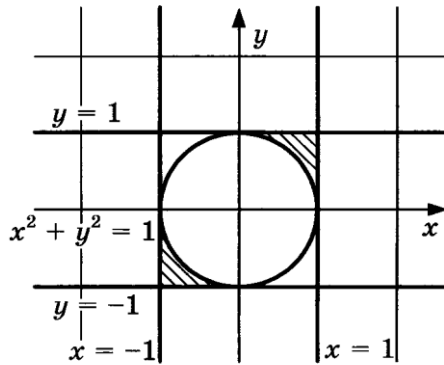
Первая строка состоит из одного символа – цифры «1». Каждая из последующих цепочек создается следующим действием: в очередную строку дважды записывается предыдущая цепочка цифр (одна за другой, подряд), а в конец приписывается еще одно число – номер строки по порядку (на i-м шаге дописывается число «i»).

Вот первые 4 строки, созданные по этому правилу:

- (1) 1
- (2) 112
- (3) 1121123
- (4) 112112311211234

Сколько раз в общей сложности встречаются в восьмой строке четные цифры (2, 4, 6, 8)?

5. Составьте программу определения принадлежности точки заштрихованной области:



Решение типовых задач: Циклы

Какое значение будет принимать переменная Y после выполнения фрагментов программы:

№	Паскаль
1	<pre> Var x,y:integer; Begin Y:=0; For x:=1 to 9 do Y:=y+1; Writeln('y=',y); End.</pre>
3	<pre> Var x,y:integer; Begin Y:=1; For x:=1 to 5 do Y:=y*x; Writeln('y=',y); End.</pre>
5	<pre> Var x,y:integer; Begin Y:=0; X:=10; While x>0 do Begin X:=x-2; Y:=y+x; End; Writeln('y=',y); End.</pre>
7	<pre> Var x,y:integer; Begin Y:=1; X:=15; repeat Y:=y*x; X:=x-3; Until x<5; Writeln('y=',y); End.</pre>
9	<pre> Var x,y:integer; Begin Y:=1; X:=10;</pre>

№	Паскаль
2	<pre> Var x,y:integer; Begin Y:=0; For x:=1 to 9 do Y:=y+x; Writeln('y=',y); End.</pre>
4	<pre> Var x,y:integer; Begin Y:=0; For x:=10 downto 5 do Y:=y+x; Writeln('y=',y); End.</pre>
6	<pre> Var x,y:integer; Begin Y:=1; X:=15; While x>5 do Begin X:=x-3; Y:=y*x; End; Writeln('y=',y); End.</pre>
8	<pre> Var x,y:integer; Begin Y:=1; X:=10; repeat Y:=y*x+x*x; X:=x-2; Until x<0; Writeln('y=',y); End.</pre>
10	<pre> Var x,y:integer; Begin Y:=0; For x:=1 to 9 do</pre>

	<pre> While x>5 do X:=x-3; Y:=y*x; Writeln('y=',y); End.</pre>
--	---

	<pre> Y:=y+1; Y:=y+x; Writeln('y=',y); End.</pre>
--	---

Решение типовых задач: Методы

Метод – это функциональный элемент класса, который реализует вычисления или другие действия, выполняемые классом или его экземпляром (объектом). Метод представляет собой законченный фрагмент кода, к которому можно обратиться по имени. Он описывается один раз, а вызываться может многократно. Совокупность методов класса определяет, что конкретно может делать класс. Например, стандартный класс `Math` содержит методы, которые позволяют вычислять значения математических функций.

Синтаксис метода:

[атрибуты] [спецификторы] тип_возвращаемого_результата имя_метода
([список_параметров])

```

{
тело_метода;
return значение
}
```

где:

1. Атрибуты и спецификторы являются необязательными элементами синтаксиса описания метода. На данном этапе атрибуты нами использоваться не будут, а из всех спецификаторов мы в обязательном порядке будем использовать спецификатор `static`, который позволит обращаться к методу класса без создания его экземпляра.
2. Тип_возвращаемого_результата определяет тип значения, возвращаемого методом. Это может быть любой тип, включая типы классов, создаваемые программистом. Если метод не возвращает никакого значения, необходимо указать тип `void` (в этом случае в теле метода отсутствует оператор `return`).
3. Имя_метода – идентификатор, заданный программистом с учетом требований, накладываемыми на идентификаторы в `C#`, отличный от тех, которые уже использованы для других элементов программы в пределах текущей области видимости.
4. Список_параметров представляет собой последовательность пар, состоящих из типа данных и идентификатора, разделенных запятыми. Параметры — это переменные или константы, которые получают значения, передаваемые методу при вызове. Если метод не имеет параметров, то список_параметров остается пустым.
5. Значение определяет значение, возвращаемое методом. Тип значения должен соответствовать типу_возвращаемого_результата или приводится к нему.

Рассмотрим простейший пример метода:

```

class Program
{
    static void Func()          //дополнительный метод
    {
        Console.Write("x= ");
        double x=double.Parse(Console.ReadLine());
        double y = 1 / x;
        Console.WriteLine("y({0})={1}", x,y );
    }

    static void Main()          //точка входа в программу
    {
        Func();                //первый вызов метода Func
    }
}
```

```

    Func();      //второй вызов метода Func
}
}

```

В данном примере в метод `Func` не передаются никакие значения, поэтому список параметров пуст. Кроме того метод ничего не возвращает, поэтому тип возвращаемого значения `void`. В основном методе `Main` мы вызвали метод `Func` два раза. Если будет необходимо, то данный метод можно будет вызвать еще столько раз, сколько потребуется для решения задачи.

Решение типовых задач: Массивы

Массив - набор элементов одного и того же типа, объединенных общим именем. Массивы в `C#` можно использовать по аналогии с тем, как они используются в других языках программирования. Однако `C#`-массивы имеют существенные отличия: они относятся к *ссылочным типам данных*, более того - реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки мусора - неиспользуемые массивы автоматически утилизируются данной системой.

Рассмотрим различные типы массивов.

Одномерные массивы

Одномерный массив - это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в `C#` начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в `C#` реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размер массива.

В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

Форма записи	Пояснения
базовый_тип [] имя_массива; Например: int [] a;	Описана ссылка на одномерный массив, которая в дальнейшем может быть использована: <ol style="list-style-type: none"> 1. для адресации на уже существующий массив; 2. передачи массива в метод в качестве параметра 3. отсроченного выделения памяти под элементы массива.
базовый_тип [] имя_массива = new базовый_тип [размер]; Например: int []a=new int [10];	Объявлен одномерный массив заданного типа и выделена память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. Элементы массива равны нулю. Замечание. Надо отметить, что в <code>C#</code> элементам массива присваиваются начальные значения по умолчанию в зависимости от базового типа. Для арифметических типов - нули, для ссылочных типов - <code>null</code> , для символов - пробел.
базовый_тип [] имя_массива={список инициализации};	Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в

Например: int []a={0, 1, 2, 3};	ссылочную переменную. Значение элементов массива соответствует списку инициализации.
------------------------------------	--

Обращения к элементам массива происходит с помощью индекса, для этого нужно указать имя массива и в квадратных скобках его номер. Например, a[0], b[10], c[i].

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Рассмотрим несколько простых примеров работы с одномерными массивами.

Пример 1.

```
static void Main()
{
    int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    int i;
    for (i = 0; i < 10; ++i)
        Console.WriteLine(myArray[i]);
}
```

Решение типовых задач:

Решение задач «Организация элементов пользовательского интерфейса».

Задачи для тренировки:

1) Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх вниз влево вправо.

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →. Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно снизу свободно
слева свободно справа свободно

Цикл **ПОКА** <условие> **команда** выполняется, пока условие истинно, иначе происходит переход на следующую строку. Сколько клеток приведенного лабиринта соответствуют требованию, что, выполнив предложенную ниже программу, РОБОТ остановится в той же клетке, с которой он начал движение?

- 1) 1 2) 0 3) 3 4) 4

НАЧАЛО

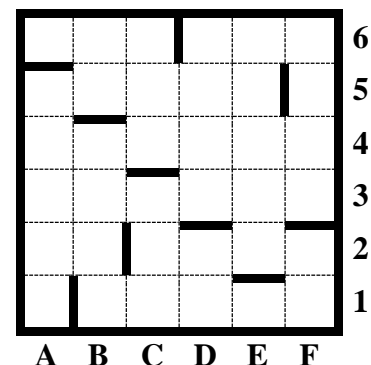
ПОКА <справа свободно> **вправо**

ПОКА <сверху свободно> **вверх**

ПОКА <слева свободно> **влево**

ПОКА <снизу свободно> **вниз**

КОНЕЦ



2) Исполнитель Черепашка перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существуют две команды:

Вперед n, где **n** – целое число, вызывающая передвижение черепашки на **n** шагов в направлении движения.

Направо m, где **m** – целое число, вызывающая изменение направления движения на **m** градусов по часовой стрелке.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в скобках повторится 5 раз.

Черепашке был дан для исполнения следующий алгоритм:

Повтори 5 [Вперед 10 Направо 72]

Какая фигура появится на экране?

1) Незамкнутая ломаная линия

2) Правильный треугольник

3) Квадрат

4) Правильный пятиугольник

3) Имеется фрагмент алгоритма, записанный на алгоритмическом языке:

n := Длина(**a**)

m := 6

b := Извлечь(**a**, **m**)

c := Извлечь(**a**, **m**-4)

b := Склеить(**b**, **c**)

c := Извлечь(**a**, **m**+2)

b := Склеить(**b**, **c**)

нц для **i** от 10 до **n**

c := Извлечь(**a**, **i**)

b := Склеить(**b**, **c**)

кц

Здесь переменные **a**, **b** и **c** - строкового типа; переменные **n**, **m**, **k** – целые. В алгоритме используются следующие функции:

Длина(x) – возвращает количество символов в строке **x**. Имеет тип «целое».

Извлечь(x,i) – возвращает **i**-й символ слева в строке **x**. Имеет строковый тип.

Склеить(x,y) – возвращает строку, в которой записаны подряд сначала все символы строки **x**, а затем все символы строки **y**. Имеет строковый тип.

Значения строк записываются в кавычках (одинарных), например **x**='школа'.

Какое значение примет переменная **b** после выполнения этого фрагмента алгоритма, если переменная **a** имела значение 'КИБЕРНЕТИКА'?

1) 'БЕРЕТ' 2) 'НИТКА' 3) 'ТИБЕТ' 4) 'НЕРКА'

4) Имеется фрагмент алгоритма, записанный на алгоритмическом языке:

m := 10

b := Извлечь(**a**, **m**)

нц для **k** от 4 до 5

c := Извлечь(**a**, **k**)

b := Склеить(**b**, **c**)

кц

нц для **k** от 1 до 3

c := Извлечь(**a**, **k**)

b := Склеить(**b**, **c**)

кц

Здесь переменные **a**, **b** и **c** - строкового типа; переменные **n**, **m**, **k** – целые. В алгоритме используются следующие функции:

Извлечь(x,i) – возвращает **i**-й символ слева в строке **x**. Имеет строковый тип.

Склеить(x,y) – возвращает строку, в которой записаны подряд сначала все символы строки **x**, а затем все символы строки **y**. Имеет строковый тип.

Значения строк записываются в кавычках (одинарных), например **x**='школа'.

Какое значение примет переменная **b** после выполнения этого фрагмента алгоритма, если переменная **a** имела значение 'ИНФОРМАТИКА'?

1) 'ФОРМАТ' 2) 'ФОРИНТ' 3) 'КОРТИК' 4) 'КОРИНФ'

5) Некий исполнитель умеет выполнять три команды:

FD<число шагов> – движение вперед на указанное число шагов

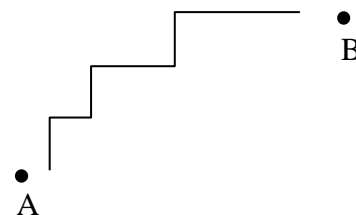
RT<число градусов> – поворот направо на указанное число градусов

REPEAT<число повторений>[<повторяющиеся действия>] – команда повторения

Например, **REPEAT 4[FD 20 RT 90]** строит квадрат со стороной 20. Какую фигуру будет представлять собой траектория движения данного исполнителя в результате выполнения команды

REPEAT 8 [FD 60 RT 45]

- 1) Равносторонний треугольник
- 2) Ромб
- 3) Правильный шестиугольник
- 4) Правильный восьмиугольник

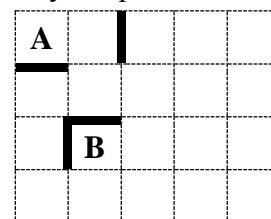


6) Некий исполнитель умеет строить лесенки. Каждая ступенька такой лесенки имеет одну единицу по высоте и целое количество единиц в длину. Одна из возможных лесенок показана на рисунке

7) Исполнитель умеет выполнять команды **ВВЕРХ** и **ВПРАВО N**, где N – длина ступеньки, причем алгоритм всегда начинается командой **ВВЕРХ** и заканчивается командой **ВПРАВО**. Необходимо, выполнив 8 команд, построить лесенку из четырех, ступенек, ведущую из точки A в точку B . Точка A имеет координаты $(0,0)$ на координатной плоскости, а точка B – координаты $(5,4)$. Сколько различных последовательностей команд могут привести к требуемому результату?

- 1) 5 2) 6 3) 3 4) 4

Исполнитель Робот действует на клетчатом поле, между соседними клетками которого могут стоять стены. Робот передвигается по клеткам поля и может выполнять следующие команды: Вверх (1), Вниз (2), Вправо (3), Влево (4).



При выполнении каждой такой команды Робот перемещается в соседнюю клетку в указанном направлении. Если же в этом направлении между клетками стоит стена, то робот разрушается.

3.Рекомендуемая литература

Основная литература:

1. Чеботарёв, С.С. Программирование на Microsoft Visual C#. Ч.1. Основы алгоритмизации и программирования [Текст]: учеб. пособие / С.С.Чеботарёв. - Челябинск: ЧОУВО МИДиС, 2018. - 88с.

Электронные издания (электронные ресурсы)

1. Казанский, А.А. Прикладное программирование на Excel 2019: учебное пособие для спо / А.А. Казанский. — 2-е изд., перераб. и доп. — Москва: Юрайт, 2020. — 171 с. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: <https://urait.ru/bcode/447551> (дата обращения: 15.09.2020).

2. Кудрина, Е.В. Основы алгоритмизации и программирования на языке C#: учебное пособие для спо / Е.В. Кудрина, М.В. Огнева. — Москва: Юрайт, 2020. — 322 с. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: <https://urait.ru/bcode/456221> (дата обращения: 15.09.2020).

3. Трофимов, В.В. Основы алгоритмизации и программирования: учебник для спо / В.В. Трофимов, Т.А. Павловская; под ред. В.В. Трофимова. — Москва: Юрайт, 2020. — 137 с. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: <https://urait.ru/bcode/454452> (дата обращения: 15.09.2020).

4. Черпаков, И.В. Основы программирования: учебник и практикум для спо / И.В. Черпаков. — Москва: Юрайт, 2020. — 219 с. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: <https://urait.ru/bcode/452182> (дата обращения: 15.09.2020).

Дополнительные источники

1. Иванова, Г.С. Программирование [Текст]: учеб. / Г.С. Иванова. - 3-е изд., стер. - М.: КНОРУС, 2014. - 432с

- . Истомин, Е.П. Информатика и программирование [Текст]: учеб. /Е.П.Истомин, С.Ю.Неклюдов, В. И. Романченко.- СПб.: Андреевский издательский дом, 2006.-248с
3. Лебедев, В.М. Программирование на VBA в MS Excel: учебное пособие для спо / В.М. Лебедев. — 2-е изд., испр. и доп. — Москва: Юрайт, 2020. — 306 с. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: <https://urait.ru/bcode/449583> (дата обращения: 15.09.2020).
4. Орлов, С.А. Теория и практика языков программирования [Текст]: учеб. для вузов / С.А.Орлов. - СПб. : Питер, 2013. - 432с.: ил. - (Учебник для вузов).
5. Павловская, Т.А. С #. Программирование на языке высокого уровня [Текст]: учеб. д / Т.А. Павловская. - СПб.: Питер, 2014. - 432с.: ил.