Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

## Факультет компьютерных систем и сетей

## Кафедра программного обеспечения информационных технологий

|  |
| --- |
|  |
|  |

# оТЧЕТ

по лабораторной работе

на тему:

### Рекурсия

|  |  |  |
| --- | --- | --- |
| Выполнил  Студент гр. 151001 |  | В. В. Лозюк |
| Проверил |  | Асс. Е.Е. Фадеева |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

#### Минск, 2022

1. Теоретические сведения по теме лабораторной работы

**Алгоритм** – система правил, четко описывающая последовательность действий, которые необходимо выполнить для решения задачи.

**Свойства правильного алгоритма**:

1. ***Дискретность*** – значение величин в каждый следующий момент времени должны получаться по определенным правилам из значений величин, имевшихся в предшествующий момент времени.
2. ***Определенность (детерминированность)*** – каждое правило алгоритма должно быть однозначным. Значения величин, получаемые в какой-то момент времени, однозначно связаны со значениями величин, вычисленных ранее.
3. ***Результативность (конечность)*** – алгоритм должен приводить к решению задачи за конечное число шагов.
4. ***Массовость*** – алгоритм должен разрабатываться в общем виде так, чтобы его можно было применить для класса задач, различающихся лишь исходными данными.

**Способы описания алгоритма:**

1. Запись на естественном языке (словесное описание)
2. Изображение в виде схемы (графическое описание)
3. Запись на алгоритмическом языке (составление программы)

**Типы алгоритмов:**

1. линейные
2. разветвляющиеся
3. циклические

**Цикл** – участок схемы, многократно повторяемый в ходе вычислений.

**Классификация циклов:**

1. По взаимному расположению:
   1. Простые;
   2. Сложные - называется цикл, если он содержит в себе другой, вложенный в него цикл;

c. Внутренние и Внешние (В теле любого оператора **цикла** могут находиться другие операторы **цикла**. При этом **цикл**, содержащий в себе другой, называют **внешним**, а **цикл**, находящийся в теле первого — **внутренним** (вложенным)).

1. По местоположению условия выполнения цикла:

**С предусловием** — цикл, который выполняется, пока истинно некоторое условие, указанное перед его началом.

**С постусловием** — цикл, в котором условие проверяется *после* выполнения тела цикла.

1. По виду условия выполнения:

а)**С параметром** — цикл, в котором переменная изменяет свое значение от начального значения до конечного с шагом, определенным заранее, и для каждого из значений переменной выполняется цикл.

б)**Итерационные** — цикл, для которого число повторений тела цикла заранее неизвестно.

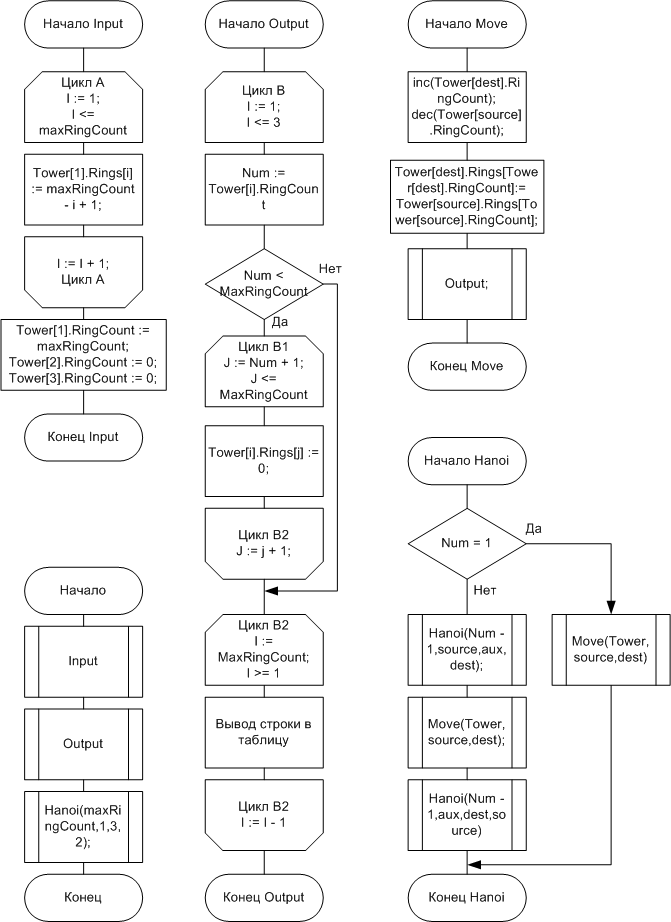
1. Задание на лабораторную работу

Ханойские башни.

1. Выполнение
   1. Разработка алгоритма

Таблица 3.1 используемые идентификаторы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя идентификатора | Назначение | Тип идентификатора | Начальное значение | Закон изменения | Имя цикла, в котором происходит изменение переменной |
| Tower | Массив, состоящий из 3х башен, с которыми идет работа | ArrOfTowers | - | - | - |
| Tower[i] | Запись, хранящая количество колец в башне с данным номером и их расположение | TTower | - | - | - |

Рисунок 3.1 – Схема работы программы

Текст программы и его описание

program HanoiskieBashni;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils;

const

maxRingCount = 4;

Type

ArrayOfRings = array[1..MaxRingCount] of integer;

TTowers = record

RingCount:integer;

Rings: ArrayOfRings;

end;

ArrOfTowers = array[1..3] of TTowers;

Var

Tower: ArrOfTowers;

Procedure Input;

var

i:integer;

begin

for i := 1 to maxRingCount do

Tower[1].Rings[i] := maxRingCount - i + 1;

Tower[1].RingCount := maxRingCount;

Tower[2].RingCount := 0;

Tower[3].RingCount := 0;

end;

Procedure Output;

var

i,j,Num: Integer;

begin

for i := 1 to 3 do

begin

Num := Tower[i].RingCount;

if Num < MaxRingCount then

begin

for j := Num + 1 to MaxRingCount do

Tower[i].Rings[j] := 0;

end;

end;

Writeln('|Башня №1|Башня №2|Башня №3|');

for i := MaxRingCount downto 1 do

Writeln('|',Tower[1].Rings[i]:4,' |',Tower[2].Rings[i]:4,' |',Tower[3].Rings[i]:4,' |');

end;

Procedure Move(var Tower:ArrOfTowers; var source:integer;var dest:integer);

begin

inc(Tower[dest].RingCount);

Tower[dest].Rings[Tower[dest].RingCount]:= Tower[source].Rings[Tower[source].RingCount];

dec(Tower[source].RingCount);

Output;

end;

Procedure Hanoi(Num: integer; source:integer; dest:integer; aux:integer);

begin

if Num = 1 then

Move(Tower,source,dest)

else

begin

Hanoi(Num - 1,source,aux,dest);

Move(Tower,source,dest);

Hanoi(Num - 1,aux,dest,source);

end;

end;

begin

Input;

Output;

Hanoi(maxRingCount,1,3,2);

readln;

end.

* 1. Тестирование и отладка программы

Таблица 3.2 Прохождение тестов программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Специфика тестирования | Номер теста | Вводимые данные | Ожидаемый результат | Полученный результат |
| Проверка на работоспособность | 1 | 3 |  | Тест пройден |

* 1. Итоговый текст программы

program HanoiskieBashni;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils;

const

maxRingCount = 3;

Type

ArrayOfRings = array[1..MaxRingCount] of integer;

TTowers = record

RingCount:integer;

Rings: ArrayOfRings;

end;

ArrOfTowers = array[1..3] of TTowers;

Var

Tower: ArrOfTowers;

Procedure Input;

var

i:integer;

begin

for i := 1 to maxRingCount do

Tower[1].Rings[i] := maxRingCount - i + 1;

Tower[1].RingCount := maxRingCount;

Tower[2].RingCount := 0;

Tower[3].RingCount := 0;

end;

Procedure Output;

var

i,j,Num: Integer;

begin

for i := 1 to 3 do

begin

Num := Tower[i].RingCount;

if Num < MaxRingCount then

begin

for j := Num + 1 to MaxRingCount do

Tower[i].Rings[j] := 0;

end;

end;

Writeln('|Башня №1|Башня №2|Башня №3|');

for i := MaxRingCount downto 1 do

Writeln('|',Tower[1].Rings[i]:4,' |',Tower[2].Rings[i]:4,' |',Tower[3].Rings[i]:4,' |');

end;

Procedure Move(var Tower:ArrOfTowers; var source:integer;var dest:integer);

begin

inc(Tower[dest].RingCount);

Tower[dest].Rings[Tower[dest].RingCount]:= Tower[source].Rings[Tower[source].RingCount];

dec(Tower[source].RingCount);

Output;

end;

Procedure Hanoi(Num: integer; source:integer; dest:integer; aux:integer);

begin

if Num = 1 then

Move(Tower,source,dest)

else

begin

Hanoi(Num - 1,source,aux,dest);

Move(Tower,source,dest);

Hanoi(Num - 1,aux,dest,source);

end;

end;

begin

Input;

Output;

Hanoi(maxRingCount,1,3,2);

readln;

end.