|  |  |
| --- | --- |
| Министерство образования Республики Беларусь | |
| Учреждение образования | |
| БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ | |
| ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | |
|  | |
|  | |
| Факультет компьютерных систем и сетей | |
| Кафедра программного обеспечения информационных технологий | |
| Дисциплина: **Название дисциплины (АББРЕВИАТУРА)** | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
| **ОТЧЁТ** | |
| по разминочной задаче № **X** | |
|  | |
| Тема работы: **Название темы** | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
| Выполнил: | Фамилия И.О. |
|  | гр. **XXXXXX** |
|  | Вариант **X** |
|  |  |
| Проверил: | Фамилия И.О. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Минск **202X** | |

содержание

[1 Постановка задачи 3](#_Toc88070774)

[2 Методика решения 4](#_Toc88070775)

[3 Структура данных 6](#_Toc88070776)

[4 Результаты расчетов 8](#_Toc88070777)

[Приложение А 9](#_Toc88070778)

[Приложение Б 12](#_Toc88070779)

[Приложение В 16](#_Toc88070780)

[Приложение Г 20](#_Toc88070781)

# Постановка задачи

Постановка задачи 1

Есть два вида бактерий: зелёные и красные. Красная бактерия за 1 такт времени становится зелёной. Зелёная бактерия делится на 2 (красную и зелёную) за 1 такт времени. Сколько бактерий, в т.ч. зелёных, красных и суммарно будет через K тактов времени, если сначала было N красных бактерий?

Вывести на экран значение зеленых, красных бактерий, их сумму на k такте.

Постановка задачи 2

Есть два вида бактерий: зёленые и красные. Красная бактерия за 1 такт времени становится зелёной. Зелёная бактерия делится на 2 (красную и зелёную) за 1 такт времени. Сколько бактерий, в т.ч. зелёных, красных и суммарно будет через K тактов времени, если сначала было Green зелёных и Red красных бактерий?

Вывести на экран количество зелёных и красных бактерий и их сумму на конечном такте.

Правила постановки:

1. Пользователь вводит количество зелёных бактерий, затем количество красных бактерий, затем количество тактов.
2. Программа учитывает некорректный ввод пользователя.
3. Количество зелёных и красных бактерий в начальный момент времени должно быть целым неотрицательным числом.
4. Количество тактов в начальный момент времени должно быть натуральным числом.
5. Программа предусматривает потенциальное переполнение переменной типа Integer при вычислениях.
6. Программа выводит количество зелёных бактерий, красных бактерий и их сумму на конечном такте.
7. Программа разделена на 3 части, каждая часть решает задачу своим способом (1 способ – через зависимость от красных и зелёных на предыдущем такте, 2 способ – через зависимость от зелёных бактерий на предыдущем и позапрошлом такте, 3 способ – через зависимость от красных бактерий на предыдущем и позапрошлом такте).

# Методика решения

Решение задачи по постановке 1

Постановка 1 отличается от постановки 2 лишь тем, что G = 0.

Решение задачи по постановке 2

Пусть заданное пользователем количество красных бактерий – R, зелёных бактерий – G. В таком случае, таблица изменения количества бактерий обоих видов будет выглядеть так:

Таблица 1 – Количество бактерий после каждого такта

|  |  |  |
| --- | --- | --- |
| Такт | Количество красных бактерий | Количество зелёных бактерий |
| 0 | R | G |
| 1 | G | R+G |
| 2 | R+G | R+2G |
| 3 | R+2G | 2R+3G |
| 4 | 2R+3G | 3R+5G |
| 5 | 3R+5G | 5R+8G |
| 6 | 5R+8G | 8R+13G |

Благодаря таблице, можно найти 3 способа решения задачи, т.е. 3 закономерности изменения количества зелёных бактерий.

Способ 1: зависимость от красных и зеленых бактерий  
на предыдущем шаге

Количество зелёных бактерий для такта I >=1 равно сумме количества красных бактерий на (I-1) такте и зелёных бактерий на (I-1) такте. Количество красных бактерий для I>=1 равно количеству зелёных бактерий на (I-1) такте.

Способ 2: зависимость от зелёных бактерий на прошлом   
и позапрошлом шаге

Количество зелёных бактерий для такта I>=2 равно сумме количества зелёных бактерий на (I-1) такте и количества зелёных на (I-2) такте. Количество красных бактерий для такта I>=1 равно количеству зелёных на (I-1) такте. При этом для 1-ого такта, для которого не существует позапрошлый такт, необходимо выполнить вычисления по первому способу.

Способ 3: зависимость от красных бактерий на прошлом   
и позапрошлом шаге

Количество зелёных бактерий для такта I>=2 равно сумме количества красных бактерий на (I-1) такте, умноженному на 2, и количества красных на (I-2) такте. Количество красных бактерий для такта I>=2 равно сумме количества красных на (I-1) такте и количества красных на (I-2) такте. При этом для 1-ого такта, для которого не существует позапрошлый такт, необходимо выполнить вычисления по первому способу.

# Структура данных

Таблица 1 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Green | Integer | Количество зелёных бактерий в начале итерации цикла |
| Red | Integer | Количество красных бактерий в конце каждой итерации |
| Tick | Integer | Количество тактов |
| SavedGreen | Integer | Количество зеленых бактерий |
| SavedRed | Integer | Количество красных бактерий |
| SavedTick | Integer | Количество пройденных тактов |
| Temp | Integer | Сохранённая сумма красных и зелёных до их модификации |
| Temp1 | Integer | Сохраняемое на чётных тактах количество зелёных бактерий |
| Temp2 | Integer | Сохраняемое на нечётных тактах количество зелёных бактерий |
| I | Integer | Параметр цикла |
| Limit | Boolean | Переменная для проверки переполнения переменных типа Integer |

# Результаты расчетов

Вследствие результатов программы на экран выводятся следующие результаты расчетов:

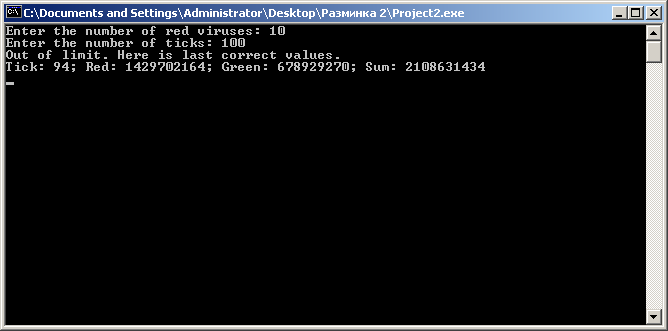


Рисунок 1 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

Зависимость от красных и зеленых на предыдущем шаге

Program Razminka2\_1;

{This program calculates viruses' number for the last correct tick.}

//Use app

{$APPTYPE CONSOLE}

//Declare modules

Uses

SysUtils;

//Declare vars

Var

Green, Red, Tick, SavedGreen, SavedRed, SavedTick, Temp, Error, I:Integer;

Input:String;

Limit:Boolean;

//Green, Red - the number of red and green viruses

//Tick - the number of ticks

//SavedGreen, SavedRed, SavedTick - last correct numbers

//Input - input string

//Temp - auxiliary argument

//Error - auxiliary operator for checking input

//I - parameter for cycle

//Limit - condition to finish the cycle

Begin

//Checking for the correct input

Repeat

Write('Enter the number of red viruses: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Red, Error);

If (Red < 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Red >= 0) And (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the number of green viruses: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Green, Error);

If (Green < 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Green >= 0) And (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the number of ticks: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Tick, Error);

If (Tick <= 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Tick > 0) And (Error = 0);

//Integer limit is not exceeded yet

Limit:= False;

SavedGreen:= 0;

SavedRed:= 0;

SavedTick:= 0;

For I:= 1 To Tick Do

Begin

//If integer limit is not exceeded yet

If (Limit = False) Then

Begin

//Searching for the new number of viruses

Temp:= Green + Red;

Red:= Green;

Green:= Temp;

//Searching for correct numbers before integer Limit

If (Green >= 0) And (Red >= 0) And ((Green + Red) >= 0) Then

Begin

SavedGreen:= Green;

SavedRed:= Red;

SavedTick:= I;

End

Else

Begin

//Integer limit was exceeded

WriteLn('Out of limit during the tick ', I, '. Here is the last correct values.');

Limit:= True;

End;

End;

End;

//Displaying the solution

WriteLn('Tick: ', SavedTick, '; Red: ', SavedRed, '; Green: ', SavedGreen, '; Sum: ', SavedRed + SavedGreen);

ReadLn;

End.

Приложение Б

(обязательное)

Исходный код программы

Зависимость от зелёных бактерий на прошлом и позапрошлом шаге

Program Razminka2\_2;

{This program calculates viruses' number for the last correct tick.}

//Use app

{$APPTYPE CONSOLE}

//Declare modules

Uses

SysUtils;

//Declare vars

Var

Green, Red, Tick, SavedGreen, SavedRed, SavedTick, Temp, Temp1, Temp2, Error, I:Integer;

Input:String;

Limit:Boolean;

//Green, Red - the number of red and green viruses

//Tick - the number of ticks

//SavedGreen, SavedRed, SavedTick - last correct numbers

//Input - input string

//Temp - auxiliary argument

//Temp1 - saved pre-pre-data for the even tick

//Temp2 - saved pre-pre-data for the odd tick

//Error - auxiliary operator for checking input

//I - parameter for cycle

//Limit - condition to finish the cycle

Begin

//Checking for the correct input

Repeat

Write('Enter the number of red viruses: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Red, Error);

If (Red < 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Red >= 0) And (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the number of green viruses: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Green, Error);

If (Green < 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Green >= 0) And (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the number of ticks: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Tick, Error);

If (Tick <= 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Tick > 0) And (Error = 0);

//Integer limit is not exceeded yet

Limit:= False;

//To prevent from random false value

SavedGreen:= 0;

SavedRed:= 0;

SavedTick:= 0;

Temp1:= 0;

Temp2:= 0;

For I:= 1 To Tick Do

Begin

//If integer limit is not exceeded yet

If (Limit = False) Then

Begin

//First tick has no pre-pre step

If I = 1 Then

Begin

Temp1:= Green;

Temp:= Green + Red;

Red:= Green;

Green:= Temp;

Temp2:= Green;

End

Else

Begin

//Even tick

If I Mod 2 = 0 Then

Begin

Red:= Green;

Green:= Temp1 + Temp2;

Temp1:= Green;

End;

//Odd tick

If I Mod 2 = 1 Then

Begin

Red:= Green;

Green:= Temp1 + Temp2;

Temp2:= Green;

End;

End;

//Searching for correct numbers before integer Limit

If (Green >= 0) And (Red >= 0) And ((Green + Red) >= 0) Then

Begin

SavedGreen:= Green;

SavedRed:= Red;

SavedTick:= I;

End

Else

Begin

//Integer Limit was exceeded

WriteLn('Out of limit during the tick ', I, '. Here is the last correct values.');

Limit:= True;

End;

End;

End;

//Displaying the solution

WriteLn('Tick: ', SavedTick, '; Red: ', SavedRed, '; Green: ', SavedGreen, '; Sum: ', SavedRed + SavedGreen);

ReadLn;

End.

Приложение В

(обязательное)

Исходный код программы

Зависимость от красных бактерий на прошлом и позапрошлом шаге

Program Razminka2\_3;

{This program calculates viruses' number for the last correct tick.}

//Use app

{$APPTYPE CONSOLE}

//Declare modules

Uses

SysUtils;

//Declare vars

Var

Green, Red, Tick, SavedGreen, SavedRed, SavedTick, Temp, Temp1, Temp2, Error, I:Integer;

Input:String;

Limit:Boolean;

//Green, Red - the number of red and green viruses

//Tick - the number of ticks

//SavedGreen, SavedRed, SavedTick - last correct numbers

//Input - input string

//Temp - auxiliary argument

//Temp1 - saved pre-pre-data for the even tick

//Temp2 - saved pre-pre-data for the odd tick

//Error - auxiliary operator for checking input

//I - parameter for cycle

//Limit - condition to finish the cycle

Begin

//Checking for the correct input

Repeat

Write('Enter the number of red viruses: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Red, Error);

If (Red < 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Red >= 0) And (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the number of green viruses: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Green, Error);

If (Green < 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Green >= 0) And (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the number of ticks: ');

ReadLn(Input);

//Error is bad symbol pos

Val(Input, Tick, Error);

If (Tick <= 0) Or (Error <> 0) Then

WriteLn('Invalid input. Enter another number.');

Until (Tick > 0) And (Error = 0);

//Integer limit is not exceeded yet

Limit:= False;

//To prevent from random false value

SavedGreen:= 0;

SavedRed:= 0;

SavedTick:= 0;

Temp1:= 0;

Temp2:= 0;

For I:= 1 To Tick Do

Begin

//If integer limit is not exceeded yet

If (Limit = False) Then

Begin

//First tick has no pre-pre step

If I = 1 Then

Begin

Temp1:= Red;

Temp:= Green + Red;

Red:= Green;

Temp2:= Red;

Green:= Temp;

End

Else

Begin

//Even tick

If I Mod 2 = 0 Then

Begin

Red:= Green;

Green:= Temp1 + 2 \* Temp2;

Temp1:= Red;

End;

//Odd tick

If I Mod 2 = 1 Then

Begin

Red:= Green;

Green:= 2 \* Temp1 + Temp2;

Temp2:= Red;

End;

End;

//Searching for correct numbers before integer Limit

If (Green >= 0) And (Red >= 0) And ((Green + Red) >= 0) Then

Begin

SavedGreen:= Green;

SavedRed:= Red;

SavedTick:= I;

End

Else

Begin

//Integer Limit was exceeded

WriteLn('Out of limit during the tick ', I, '. Here is the last correct values.');

Limit:= True;

End;

End;

End;

//Displaying the solution

WriteLn('Tick: ', SavedTick, '; Red: ', SavedRed, '; Green: ', SavedGreen, '; Sum: ', SavedRed + SavedGreen);

ReadLn;

End.

Приложение Г

(обязательное)

Тестовые наборы

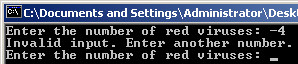
Тестовая ситуация 1: проверка ввода данных

Тестовая ситуация: проверка, сообщит ли программа о некорректном вводе, если введено отрицательное или дробное число.

Исходные данные: Red = -4.

Ожидаемый результат: Invalid input. Enter another number.

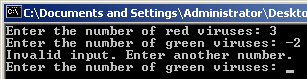
Полученный результат:



Исходные данные: Green = -2.

Ожидаемый результат: Invalid input. Enter another number.

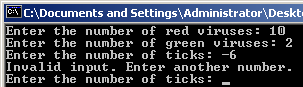
Полученный результат:



Исходные данные: Ticks = -6.

Ожидаемый результат: Invalid input. Enter another number.

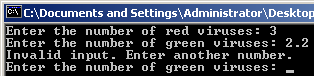
Полученный результат:



Исходные данные: Green = 2.2.

Ожидаемый результат: Invalid input. Enter another number.

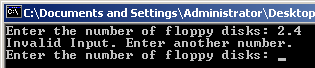
Полученный результат:



Исходные данные: Red = 2.4.

Ожидаемый результат: Invalid input. Enter another number.

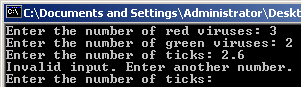
Полученный результат:



Исходные данные: Ticks = 2.6.

Ожидаемый результат: Invalid input. Enter another number.

Полученный результат:



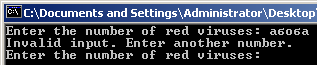
Тестовая ситуация 2: проверка ввода данных

Тестовая ситуация: проверка, сообщит ли программа о некорректном вводе, если введено буквенное значение.

Исходные данные: Red = абоба.

Ожидаемый результат: Invalid input. Enter another number.

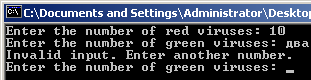
Полученный результат:



Исходные данные: Green = два.

Ожидаемый результат: Invalid input. Enter another number.

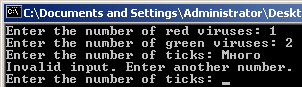
Полученный результат:



Исходные данные: Ticks = много.

Ожидаемый результат: Invalid input. Enter another number.

Полученный результат:



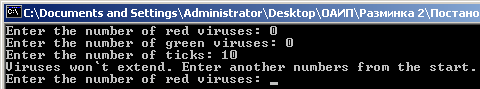
Тестовая ситуация 3: проверка возможности вычислений

Тестовая ситуация: проверка, сообщит ли программа о невозможности вычислений, если задача с заданными исходными данными не имеет решения.

Исходные данные: Red = 0, Green = 0 (бактерии не делятся).

Ожидаемый результат: Viruses won`t extend. Enter another numbers from the start.

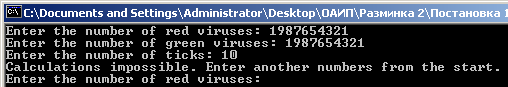
Полученный результат:



Исходные данные: Red = 1987654321, Green = 1987654321 (превышение лимита до начала вычислений).

Ожидаемый результат: Calculations impossible. Enter another numbers from the start.

Полученный результат:



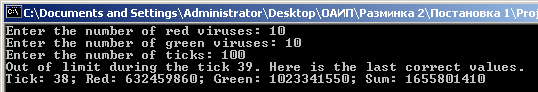
Тестовая ситуация 3: проверка лимита

Тестовая ситуация: проверка, выведет ли программа последнее корректное значение, если впоследствии лимит integer был превышен.

Исходные данные: Red = 10; Green = 20; Ticks = 100;

Ожидаемый результат: Out of limit. Here is last correct values.

Полученный результат:



Тестовая ситуация 4: проверка вычислений

Тестовая ситуация для проверки вывода правильных решений при количестве красных бактерий, равном двум, и количестве тиков, равном трём.

Исходные данные: Red = 10; Green = 20; Ticks = 30

Ожидаемый результат: 21783090 красных, 35245780 зелёных и 57028870 в сумме.

Полученный результат:

