Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по лабораторной работе №4

Тема работы: Работа с одномерными массивами

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2022

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc120912705)

[2 Методика решения 4](#_Toc120912706)

[2.1 Описание оператора try..except..end 4](#_Toc120912707)

[2.2 Условия ввода 4](#_Toc120912708)

[2.3 Проверка введенных данных 4](#_Toc120912709)

[2.3.1 Проверка ввода количества цифр 4](#_Toc120912710)

[2.3.2 Проверка корректности введенного числа 4](#_Toc120912711)

[2.4 Краткое описание алгоритма решения задачи 5](#_Toc120912712)

[3 Текстовый алгоритм решения задачи 6](#_Toc120912713)

[4 Структура данных 7](#_Toc120912714)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 8](#_Toc120912715)

[6 Результаты расчетов 10](#_Toc120912716)

[Приложение А 11](#_Toc120912717)

[Приложение Б 15](#_Toc120912718)

[Приложение С 18](#_Toc120912719)

[С.1 Код с использованием сортировки, используя генератор псевдослучайных чисел 18](#_Toc120912720)

[С.2 Результаты расчётов 21](#_Toc120912721)

[Приложение D 22](#_Toc120912722)

[D.1 Исходный код, используя типизированный массив 22](#_Toc120912723)

[D.2 Результаты расчётов 24](#_Toc120912724)

[Приложение E 25](#_Toc120912725)

[E.1 Исходный код, используя константный массив 25](#_Toc120912726)

[E.1 Результаты расчётов 27](#_Toc120912727)

# Постановка задачи

Вводится произвольная последовательность чисел *a1, a2, …, am*. Найти частоту повторения каждого числа. Результаты расчетов поместить в одномерный массив В, где каждая пара элементов (*Bi, Bi+1*) соответственно равна: *Bi* – различные элементы последовательности *a1, a2, …, am*, а *Bi+1* – число их повторений. Количество пар (*Bi, Bi+1*) соответствует числу различных чисел, входящих в последовательность *a1, a2, …, am*.

# [Методика решения](#_Toc83996305)

## Описание оператора try..except..end

Оператор try..except..end имеет вид:

try  
 операторы;  
except  
  блок обработки исключений;  
end;

Выполнение блока начинается с секции try, при отсутствии исключительных ситуаций только она и выполняется. Секция except получает управление в случае возникновения исключительной ситуации. После обработки выполняются операторы, стоящие после end.

## Условия ввода

Условия ввода:

* количество чисел в последовательности должно быть больше 1 и не больше 100;
* числа в последовательности должны быть целочисленными;

## Проверка введенных данных

### Проверка ввода количества цифр

Проверка ввода количества цифр происходит с помощью цикла с предусловием repeat..until, чтобы при вводе некорректных данных пользователь заново заполнял их. В теле цикла с помощью оператора try..except..end (описание оператора см. [главу 2.1](#_Описание_оператора_try..except..end)) проверяем целочисленность; оператором if проверяет принадлежность заданному промежутку.

### Проверка корректности введенного числа

Проверка корректности введенного числа происходит с помощью цикла с предусловием repeat..until, чтобы при вводе некорректных данных пользователь заново заполнял их. В теле цикла с помощью оператора try..except..end (описание оператора см. [главу 2.1](#_Описание_оператора_try..except..end)) проверяем целочисленность.

## Краткое описание алгоритма решения задачи

В данном алгоритме решения нужно сделать следующие действия:

1. Первый элемент массива А добавляем в массив Б по индексу i := 1 (где i – текущий индекс в массиве B). Также в массив Б по индексу i + 1 присваеваем значение 1.
2. Проходимся по массиву A со второго индекса до SizeA.
3. Проверяем, есть ли текущий элемент массива А в массиве B по индексам i..SizeB с шагом 2. Если есть, добавляем единицу к массиву B по индексу i+1. Иначе размер SizeB увеличиваем на 2, добавляем текущее число в массив В по индексу SizeB, также в массив Б по индексу SizeB +1 присваеваем значение 1.
4. Увеличиваем текущий индекс в массиве В i на два и возвращаемся в пункт 2.

# Текстовый алгоритм решения задачи

Таблица 1 – Алгоритм решения

|  |  |
| --- | --- |
| Номер  шага | Назначение шага |
|  | Ввод Size |
|  | Ввод InputNum[1..Size] |
|  | LastPosRes:= 1 |
|  | Res[LastPosRes]:= InputNum[1] |
|  | Res[LastPosRes+1]:= 1 |
|  | i:= 2 |
|  | Начало цикла А1. Проверка выполнения условия (i <= Size). Если условие истинно, перейти к шагу 8, иначе – к шагу 22 |
|  | FindNumInRes:= False |
|  | j:= 1 |
|  | Начало цикла А1.1. Проверка выполнения условий (j <= LastPosRes) и (FindNumInRes = False). Если условия истинны, перейти к шагу 11, иначе – к шагу 16 |
|  | Если условие InputNum[i] = Res[j] истинно, перейти к шагу 12. Иначе перейти к шагу 14 |
|  | Res[j+1]:= Res[j+1] + 1 |
|  | FindNumInRes:= True |
|  | j:= j + 2 |
|  | Конец цикла А1.1. Вернуться к шагу 10 |
|  | Если условие FindNumInRes = False истинно, перейти к шагу 17. Иначе перейти к шагу 20 |
|  | LastPosRes:= LastPosRes + 2 |
|  | Res[LastPosRes]:= InputNum[i] |
|  | Res[LastPosRes+1]:= 1 |
|  | i:= i + 1 |
|  | Конец цикла А1. Вернуться к шагу 7 |
|  | Вывод Res[1..LastPosRes] |
|  | Останов. |

# Структура данных

Таблица 2 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| MaxSize | Integer | Максимальное допустимое количество чисел в последовательности |
| InputNum | Array [1..MaxSize] of Integer | Массив введенных чисел |
| Res | Array [1..MaxSize\*2] of Integer | Массив, который хранит частоту повторения чисел из массива введенных чисел |
| Size | Integer | Количество чисел в последовательности |
| i | Integer | Счетчик цикла |
| j | Integer | Счетчик цикла |
| LastPosRes | Integer | Последняя позиция в массиве, который хранит частоту повторения чисел |
| flag | Boolean | Индикатор проверки правильности ввода |

# Схема алгоритма решения задачи по ГОСТ 19.701-90



Рисунок 1 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 1)



Рисунок 2 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 2)

# Результаты расчетов

После запуска программы на экран выводятся следующие результаты расчетов:

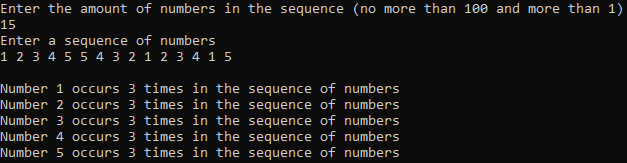


Рисунок 3 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

Program OnePass;

{

Enter a sequence of numbers,

find the frequency of repetition of each number.

Put the results into a one-dimensional array

}

{$APPTYPE CONSOLE}

Uses

System.SysUtils;

Const

MaxSize= 100;

//MaxSize - maximum amount of numbers

Var

InputNum: Array [1..MaxSize] of Integer;

Res: Array [1..MaxSize\*2] of Integer;

Size, i, j, LastPosRes: integer;

flag, FindNumInRes: boolean;

//InputNum - array of entered numbers

//Res - array of frequency of repetition of each number

//Size - size of used array

//i, j - cycle counter

//LastPosRes - last position (where the numbers are stored)

//in Res

//Shift - amount of shifts

//flag - flag to confirm the correctness of entering numbers

//FindNumInRes - indicator of finding the current number in

//Res

Begin

Writeln('Enter the amount of numbers in the sequence (no

more than ',MaxSize,' and more than 1)');

//Cycle with postcondition for entering correct data.

Repeat

//Initialize the flag

flag:= False;

//Validating the correct input data type

Try

Readln(Size);

Except

Writeln('Wrong input of amount of numbers! It must be

an integer');

flag:= True;

End;

//Validate Range

if ((Size > MaxSize) or (Size <= 1)) and (flag = False)

then

begin

Writeln('Wrong input of amount of numbers! It must be

>1 and <=',MaxSize);

flag:= True;

end;

Until flag = False;

Writeln('Enter a sequence of numbers');

//Cycle with a parameter to write a sequence of numbers

//into an array. Cycle with postcondition for entering

//correct data.

for i := 1 to Size do

repeat

//Initialize the flag

flag:= False;

//Validating the correct input data type

Try

Read(InputNum[i]);

Except

Writeln('The last number is entered incorrectly! Keep

entering numbers');

flag:= True;

End;

until flag = False;

//Initializing variables for a cycle

LastPosRes:= 1;

Res[LastPosRes]:= InputNum[1];

Res[LastPosRes+1]:= 1;

//Iterate all the numbers in sequence from the 2nd number,

//since the 1st number is already written

for i := 2 to Size do

begin

//Initializing variables for a cycle

FindNumInRes:= False;

j:= 1;

//Iterate current exist numbers in Res

while (j <= LastPosRes) and (FindNumInRes = False) do

begin

//If the number exists in Res, then write it down

//Exiting the cycle

if InputNum[i] = Res[j] then

begin

Res[j+1]:= Res[j+1] + 1;

FindNumInRes:= True;

end;

//Modernize j

j:= j + 2;

end;

//If it does not exist in Res, then create the current

//number in it

if FindNumInRes = False then

begin

LastPosRes:= LastPosRes + 2;

Res[LastPosRes]:= InputNum[i];

Res[LastPosRes+1]:= 1;

end;

end;

Writeln;

//Output the number and frequency of repetition

i:= 1;

while i <= LastPosRes do

begin

Writeln('Number ',Res[i],' occurs ',Res[i+1],' times in

the sequence of numbers');

i:= i + 2;

end;

Readln(i);

End.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

Тест 1

Исходные данные: Некорректный ввод количества чисел в последовательности

Ожидаемый результат: Повторная попытка

Полученный результат:

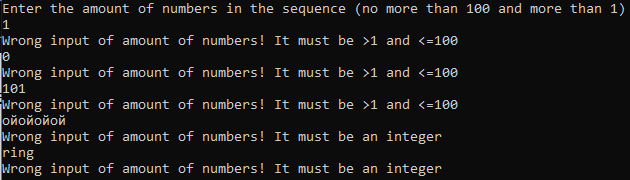


Рисунок 4 – Результаты расчетов

Тест 2

Исходные данные: Некорректный ввод чисел в последовательности

Ожидаемый результат: Повторная попытка

Полученный результат:

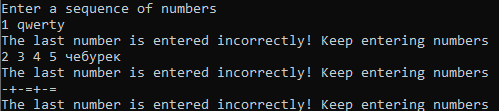


Рисунок 5 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

Таблица 3 - Результаты расчётов

|  |  |  |
| --- | --- | --- |
| Тест | Исходные данные и полученный результат | Проверка полученного результата |
| 1. 1. |  | Число 1 повторяется 3 раза  Число 2 повторяется 3 раз  Число 3 повторяется 3 раза  Число 4 повторяется 3 раз  Число 5 повторяется 3 раз  Количество повторений каждого числа в последовательности совпадают с полученным результатом |
| 1. 2. |  | Число 1 повторяется 2 раза  Число 2 повторяетсся 2 раз  Число 3 повторяется 2 раз  Число 4 повторяется 2 раз  Число 5 повторяется 2 раз  Число 6 повторяется 1 раза  Число 6 повторяетсся 1 раз  Число 7 повторяется 1 раз  Число 8 повторяется 1 раз  Число 9 повторяется 1 раз  Число 10 повторяется 1 раза  Число 11 повторяетсся 1 раз  Число 12 повторяется 1 раз  Число 13 повторяется 1 раз  Число 14 повторяется 1 раз  Число 15 повторяется 1 раз  Количество повторений каждого числа последовательности совпадают с полученным результатом |
|  |  | Число -2 повторяется 3 раза  Число -3 повторяется 2 раза  Число 2 повторяется 3 раз  Число 3 повторяется 2 раза  Количество повторений каждого числа последовательности совпадают с полученным результатом |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| 4. |  | Число -300 повторяется 1 раза  Число 300 повторяется 2 раза  Число 100 повторяется 2 раз  Число -100 повторяется 1 раза  Число -500 повторяется 3 раза  Число 1 повторяется 2 раза  Число 3 повторяется 2 раза  Число 5 повторяется 2 раза  Количество повторений каждого числа последовательности совпадают с полученным результатом |
| 5. |  | Число 12345 повторяется 2 раза  Число 54321 повторяется 1 раза  Число 555 повторяется 2 раз  Количество повторений каждого числа последовательности совпадают с полученным результатом |
| 6. |  | Число -17 повторяется 2 раза  Число -33333 повторяется 1 раза  Число -3333 повторяется 1 раз  Число -333 повторяется 1 раз  Число -33 повторяется 1 раз  Число -3 повторяется 1 раз  Количество повторений каждого числа последовательности совпадают с полученным результатом |
| 7. |  | Число 7 повторяется 20 раза  Количество повторений каждого числа последовательности совпадают с полученным результатом |

Приложение С

## С.1 Код с использованием сортировки, используя генератор псевдослучайных чисел

Program UsingSort;

{

Enter a sequence of numbers,

find the frequency of repetition of each number.

Put the results into a one-dimensional array

}

{$APPTYPE CONSOLE}

Uses

System.SysUtils;

Const

MaxSize= 100;

//MaxSize - maximum amount of numbers

Var

InputNum: Array [0..MaxSize] of Integer;

Res: Array [1..MaxSize\*2] of Integer;

Size, i, j, LastPosRes: integer;

flag: boolean;

//InputNum - array of numbers

//Res - array of frequency of repetition of each number

//Size - size of used array

//i,j - cycle counter

//LastPosRes - last position (where the numbers are stored)

//in Res

//Shift - amount of shifts

//flag - flag to confirm the correctness of entering numbers

Begin

Writeln('Enter the amount of numbers in the sequence (no

more than ',MaxSize,' and more than 1)');

//Cycle with postcondition for entering correct data.

Repeat

//Initialize the flag

flag:= False;

//Validating the correct input data type

Try

Readln(Size);

Except

Writeln('Wrong input of amount of numbers! It must be

an integer');

flag:= True;

End;

//Validate Range

if ((Size > MaxSize) or (Size <= 1)) and (flag = False)

then

begin

Writeln('Wrong input of amount of numbers! It must be

>1 and <=',MaxSize);

flag:= True;

end;

Until flag = False;

Writeln('Subsequence:');

//Generate pseudo-random numbers from -10 to 10

Randomize;

for i := 1 to Size do

begin

InputNum[i]:= Random(20) - 10;

Write(InputNum[i],' ');

end;

//Sort array InputNum using insertion method with barrier

//element

for i:= 2 to Size do

begin

//Adding the current checked number to the barrier

//element

InputNum[0]:= InputNum[i];

//Initialize j for the cycle

j:=i-1;

//Change numbers until all elements are sorted up to the

//current number

while (j>=1) and (InputNum[j] > InputNum[0]) do

begin

InputNum[j+1]:=InputNum[j];

j:=j-1

end;

//Assign the number reached in the cycle to the barrier

//element

InputNum[j+1]:= InputNum[0];

end;

//Initializing variables for a cycle

LastPosRes:= 1;

Res[LastPosRes]:= InputNum[1];

Res[LastPosRes+1]:= 1;

//Iterate all the numbers in sequence from the 2nd number,

//since the 1st number is already written

for i := 2 to Size do

begin

//If current number = current value in Res, increase

//frequency of repetition of a number

if InputNum[i] = Res[LastPosRes] then

Res[LastPosRes+1]:= Res[LastPosRes+1] + 1

//Else create a new number in Res for which will look for

//the repetition frequency

else

begin

LastPosRes:= LastPosRes + 2;

Res[LastPosRes]:= InputNum[i];

Res[LastPosRes+1]:= 1;

end;

end;

Writeln;

//Output the number and frequency of repetition

i:= 1;

while i <= LastPosRes do

begin

Writeln('Number ',Res[i],' occurs ',Res[i+1],' times in

the sequence of numbers');

i:= i + 2;

end;

Readln(i);

End.

## С.2 Результаты расчётов

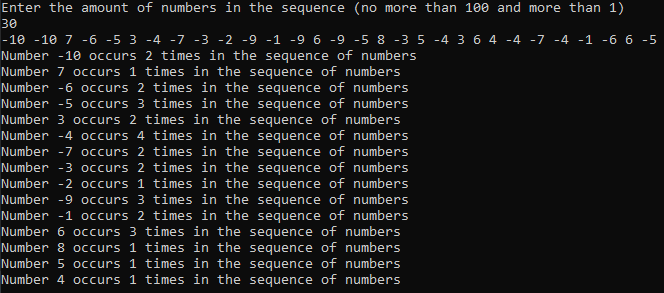


Рисунок 6 – Результаты расчетов

Приложение D

## D.1 Исходный код, используя типизированный массив

Program OnePassTypeVariable;

{

Enter a sequence of numbers,

find the frequency of repetition of each number.

Put the results into a one-dimensional array

}

{$APPTYPE CONSOLE}

Uses

System.SysUtils;

Var

InputNum: Array [1..10] of Integer = (1, 2, 2, 1, 2, 3, 4,

11, 12, 3);

Res: Array [1..10\*2] of Integer;

Size, i, j, LastPosRes: integer;

flag, FindNumInRes: boolean;

//InputNum - array of numbers

//Res - array of frequency of repetition of each number

//Size - size of used array

//i, j - cycle counter

//LastPosRes - last position (where the numbers are stored)

//in Res

//Shift - amount of shifts

//flag - flag to confirm the correctness of entering numbers

//FindNumInRes - indicator of finding the current number in

//Res

Begin

//Determine the size of array

Size:= length(InputNum);

//Outputting a sequence

Writeln('Subsequence:');

for i := 1 to Size do

Write(InputNum[i],' ');

//Initializing variables for a cycle

LastPosRes:= 1;

Res[LastPosRes]:= InputNum[1];

Res[LastPosRes+1]:= 1;

//Iterate all the numbers in sequence from the 2nd number,

//since the 1st number is already written

for i := 2 to Size do

begin

//Initializing variables for a cycle

FindNumInRes:= False;

j:= 1;

//Iterate current exist numbers in Res

while (j <= LastPosRes) and (FindNumInRes = False) do

begin

//If the number exists in Res, then write it down

//Exiting the cycle

if InputNum[i] = Res[j] then

begin

Res[j+1]:= Res[j+1] + 1;

FindNumInRes:= True;

end;

//Modernize j

j:= j + 2;

end;

//If it does not exist in Res, then create the current

//number in it

if FindNumInRes = False then

begin

LastPosRes:= LastPosRes + 2;

Res[LastPosRes]:= InputNum[i];

Res[LastPosRes+1]:= 1;

end;

end;

Writeln;

//Output the number and frequency of repetition

i:= 1;

while i <= LastPosRes do

begin

Writeln('Number ',Res[i],' occurs ',Res[i+1],' times in

the sequence of numbers');

i:= i + 2;

end;

Readln(i);

End.

## D.2 Результаты расчётов

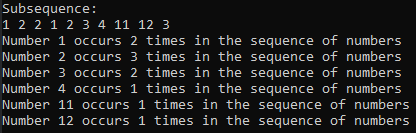


Рисунок 7 – Результаты расчетов

Приложение E

## E.1 Исходный код, используя константный массив

Program OnePassTypeConst;

{

Enter a sequence of numbers,

find the frequency of repetition of each number.

Put the results into a one-dimensional array

}

{$APPTYPE CONSOLE}

Uses

System.SysUtils;

Const

InputNum: Array [1..10] of Integer = (1, 2, 3, 4, 1, 0, 2,

0, 1, 4);

//InputNum - array of numbers

Var

Res: Array [1..10\*2] of Integer;

Size, i, j, LastPosRes: integer;

flag, FindNumInRes: boolean;

//Res - array of frequency of repetition of each number

//Size - size of used array

//i, j - cycle counter

//LastPosRes - last position (where the numbers are stored)

//in Res

//Shift - amount of shifts

//flag - flag to confirm the correctness of entering numbers

//FindNumInRes - indicator of finding the current number in

//Res

Begin

//Determine the size of array

Size:= length(InputNum);

//Outputting a sequence

Writeln('Subsequence:');

for i := 1 to Size do

Write(InputNum[i],' ');

//Initializing variables for a cycle

LastPosRes:= 1;

Res[LastPosRes]:= InputNum[1];

Res[LastPosRes+1]:= 1;

//Iterate all the numbers in sequence from the 2nd number,

//since the 1st number is already written

for i := 2 to Size do

begin

//Initializing variables for a cycle

FindNumInRes:= False;

j:= 1;

//Iterate current exist numbers in Res

while (j <= LastPosRes) and (FindNumInRes = False) do

begin

//If the number exists in Res, then write it down

//Exiting the cycle

if InputNum[i] = Res[j] then

begin

Res[j+1]:= Res[j+1] + 1;

FindNumInRes:= True;

end;

//Modernize j

j:= j + 2;

end;

//If it does not exist in Res, then create the current

//number in it

if FindNumInRes = False then

begin

LastPosRes:= LastPosRes + 2;

Res[LastPosRes]:= InputNum[i];

Res[LastPosRes+1]:= 1;

end;

end;

Writeln;

//Output the number and frequency of repetition

i:= 1;

while i <= LastPosRes do

begin

Writeln('Number ',Res[i],' occurs ',Res[i+1],' times in

the sequence of numbers');

i:= i + 2;

end;

Readln(i);

End.

## E.1 Результаты расчётов

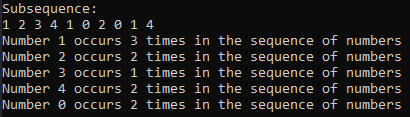


Рисунок 8 – Результаты расчетов