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 №9

Тема работы: Множества

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2023

Содержание

[1 Постановка задачи 4](#_Toc136805479)

[2 Методика решения 5](#_Toc136805480)

[3 Описание алгоритмов решения задачи 6](#_Toc136805481)

[4 Структура данных 7](#_Toc136805482)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 9](#_Toc136805483)

[5.1 Схема алгоритма GenerateRandomGroup 10](#_Toc136805484)

[5.2 Схема алгоритма OutputUnvisited 12](#_Toc136805485)

[5.3 Схема алгоритма IsNowhere 13](#_Toc136805486)

[6 Результаты расчетов 16](#_Toc136805487)

[Приложение А 17](#_Toc136805488)

[Приложение Б 21](#_Toc136805489)

# Постановка задачи

Заданы:

Type

Name = (Vasya, Volodya, Ira, Lida, Marina, Misha,

Natasha, Oleg, Olga, Sveta, Julia);

Guests = set of Name;

Group = array[Name] of Guests;

Описать логическую функцию нигде (Gr), определяющую, есть ли в группе Gr типа Group хотя бы один человек, не побывавший в гостях ни у одного человека из группы и указать его имя (Gr[x] - множество людей, бывших в гостях у человека с именем X; X не принадлежит множеству Gr[x]).

# Методика решения

Множество в языке Delphi представляет собой коллекцию уникальных элементов одного типа. В Delphi множество объявляется с помощью ключевого слова set of, за которым следует указание базового типа элементов множества. Множество в Delphi может содержать не более 256 элементов.

В данной задаче множества были использованы для представления информации о гостях, которых каждый человек из группы посещал. Был определен тип TGuests, который представляет множество имен из перечислимого типа TName.

Затем был объявлен тип TGroup, который представляет массив, где каждому человеку из перечислимого типа TName соответствует множество его посещенных гостей. Таким образом, каждый элемент массива TGroup является множеством типа TGuests.

Для нахождения людей, которые не побывали ни у кого в гостях использовались множества для проверки, был ли каждый гость посещен хотя бы одним человеком. Путем перебора каждого гостя и каждого человека в группе проверялось, есть ли гость в множестве гостей человека. Если гость был посещен, соответствующий элемент в логическом массиве устанавливался True. Это позволяло определить, есть ли люди в группе, которые не посещали никого.

Таким образом, множества в языке Delphi были применены в данной задаче для представления гостей, посещенных каждым человеком в группе, и для определения, есть ли люди, которые не посещали никого.

# Описание алгоритмов решения задачи

Таблица 1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Название алгоритма | Формальные  параметры | Предпола-гаемый тип реализации |
| 1 | Основной алгоритм | Вызов следующих подпрограмм:  IsNowhere, OutputUnvisited, GenerateRandom-Group |  |  |
| 2 | GenerateRan-domGroup  (  AGr  ) | Генерирует случайным образом группу AGr | AGr – получает от фактического параметра адрес с защитой | Проце-дура |
| 3 | OutputUnvisited  (  AVisited  ) | Выводит людей, которе не побывали ни у кого в гостях, используя AVisited | AVisited – получает от фактического параметра адрес с защитой | Проце-дура |
| 4 | IsNowhere  (  AGr,  AVisited,  Res  ) | Находит людей из группы AGr, которые не побывали ни у кого в гостях, и записывает в AVisited. Результат записывает в Res | AGr – получает от фактического параметра адрес с защитой.  AVisited – получает от фактического параметра адрес.  Res – получает от фактического параметра адрес,  возвращаемый параметр | Функция. Res – возвра-щаемый функцией параметр |

# Структура данных

Таблица 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TName | (Vasya, Volodya, Ira, Lida, Marina, Misha, Natasha, Oleg, Olga, Sveta, Julia) | Перечислимый скалярный тип. Представляет имена людей в группе |
| TGuests | Set of TName | Множество людей, которые побывали в гостях |
| TGroup | Array[TName] of TGuests | Представляет группу людей и информацию о том, какие люди были в гостях |
| TVisited | Array[TName] of Boolean | Используется для отслеживания информации о том, был ли каждый человек в гсотях |

Таблица 3 – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Gr | TGroup | Представляет группу людей и информацию о том, какие люди были в гостях |
| Visited | TVisited | Используется для отслеживания информации о том, был ли каждый человек в гсотях |

Таблица 4 – Структура данных алгоритма GenerateRandomGroup(AGr)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | | Тип параметра |
| AGr | TGroup | Представляет группу людей и информацию о том, какие люди были в гостях | | Формальный |
| Person | TName | Текущий человек, которых принимает гостей | Локальный | |
| Guest | TName | Текущий гость, который посещал других людей | Локальный | |

Таблица 5 – Структура данных алгоритма OutputUnvisited(AVisited)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | | Тип параметра |
| AVisited | TVisited | Используя данный массив, выводит людей людей, которые не побывали ни у кого в гостях | | Формальный |
| Person | TName | Текущий гость, который посещал других людей | Локальный | |

Таблица 6 – Структура данных алгоритма IsNowhere(AGr, AVisited, Res)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | | Тип параметра |
| AGr | TGroup | Представляет группу людей и информацию о том, какие люди были в гостях | Формальный | |
| AVisited | TVisited | Используется для отслеживания информации о том, был ли каждый человек в гсотях | Формальный | |
| Res | Boolean | Индикатор того, были ли все люди из группы у кого-нибудь в гостях | Формальный | |
| Person | TName | Текущий человек, которых принимает гостей | Локальный | |
| Guest | TName | Текущий гость, который посещал других людей | Локальный | |
| flag | Boolean | Индикатор того, был ли гость у кого-нибудь в гостях | Локальный | |

# Схема алгоритма решения задачи по ГОСТ 19.701-90



Рисунок 1 – Схема алгоритма в соответствии с ГОСТ 19.701-90

## Схема алгоритма GenerateRandomGroup



Рисунок 2 – Схема алгоритма GenerateRandomGroup (часть 1)



Рисунок 3 – Схема алгоритма GenerateRandomGroup (часть 2)

## Схема алгоритма OutputUnvisited



Рисунок 4 – Схема алгоритма OutputUnvisited

## Схема алгоритма IsNowhere



Рисунок 5 – Схема алгоритма IsNowhere (часть 1)



Рисунок 6 – Схема алгоритма IsNowhere (часть 2)



Рисунок 7 – Схема алгоритма IsNowhere (часть 3)

# Результаты расчетов



Рисунок 8 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

program Lab4;

{

The program generates a random group of people, where each

person has a list of guests they have visited. It then checks

if there are individuals who have not visited anyone and

outputs their names. If everyone in the group has visited

someone, it displays a message

indicating so.

}

type

TName = (Vasya, Volodya, Ira, Lida, Marina, Misha,

Natasha, Oleg, Olga, Sveta, Julia);

TGuests = set of TName;

TGroup = array[TName] of TGuests;

TVisited = array[TName] of Boolean;

// TName - enumeration representing the names of

// individuals in the group

// TGuests - set of names representing the guests visited

// by an individual

// TGroup - array representing the group, where each indi

// vidual is associated with their visited guests

// TVisited - array indicating whether an individual has

// visited someone or not

const

Name: array[TName] of string = ('Vasya', 'Volodya', 'Ira',

'Lida', 'Marina', 'Misha', 'Natasha', 'Oleg',

'Olga', 'Sveta', 'Julia');

// Name - array containing the names corresponding to each

// TName enumeration value

// Finds people who haven't visited anyone

function IsNowhere(const AGr: TGroup; var AVisited:

TVisited): Boolean;

var

Person: TName;

Guest: TName;

flag: Boolean;

begin

// Iterate through each guest and check if they have been

// visited by any individual in the group

for Guest := Low(TName) to High(TName) do

begin

flag:= true;

Person := Low(TName);

while (Person <= High(TName)) and flag do

begin

// If the guest has been visited by the current

// individual, mark them as visited

if Guest in AGr[Person] then

begin

AVisited[Guest] := True;

flag := False;

end;

Inc(Person);

end;

end;

// Iterate through each individual and check if they have

// not visited anyone

Result:= False;

Person := Low(TName);

while (Person <= High(TName)) and not Result do

begin

// If an individual has not visited anyone, set the Result

// flag to True

if not AVisited[Person] then

Result:= True;

Inc(Person);

end;

end;

procedure OutputUnvisited(const AVisited: TVisited);

var

Person: TName;

begin

// Output the names of individuals who have not visited

// anyone

for Person := Low(TName) to High(TName) do

if not AVisited[Person] then

Writeln(Name[Person], ' ');

end;

procedure GenerateRandomGroup(var AGr: TGroup);

var

Person: TName;

Guest: TName;

begin

Randomize;

for Person := Low(TName) to High(TName) do

begin

// Output the current individual for whom we are

// generating the guest list

Writeln('People who visited ', Name[Person], ':');

// Clear the set of guests for the current individual

AGr[Person] := [];

for Guest := Low(TName) to High(TName) do

begin

// Randomly include guests for the current individual

if (Random(10) = 0) and (Guest <> Person) then

begin

Include(AGr[Person], Guest);

Write(Name[Guest], ' ');

end;

end;

// If there are no guests for the current individual,

// output "None"

if AGr[Person] = [] then

Write('None');

Writeln;

end;

end;

var

Gr: TGroup;

Visited: TVisited;

begin

// Generate a random group with guest lists for each

// individual

GenerateRandomGroup(Gr);

Writeln;

// Check if there are individuals who have not visited

// anyone and output their names

if IsNowhere(Gr, Visited) then

begin

Writeln('The following people have not visited

someone:');

OutputUnvisited(Visited);

end

// If everyone in the group has visited someone, output a

// message indicating so

else

Writeln('All the people in the group have been visiting

someone');

Readln;

end.

Приложение Б

(обязательное)

Тестовые наборы

Тест 1



Рисунок 9 – Результаты расчетов

Тест 2



Рисунок 10 – Результаты расчетов

Тест 3



Рисунок 11 – Результаты расчетов

Тест 4



Рисунок 12 – Результаты расчетов

Тест 5



Рисунок 13 – Результаты расчетов