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 № 8

Тема работы: Множества

Выполнил

студент: гр.551004 Довыдёнок М.А.

Проверил: Фадеева Е.П.

Минск 2016

Содержание

[1 Постановка задачи 3](#_Toc450247434)

[2 Описание алгоритмов 4](#_Toc450247435)

[3 Структура типов 5](#_Toc450247436)

[3.1 Структура типов основной программы 5](#_Toc450247437)

[4 Структура данных 6](#_Toc450247438)

[4.1 Структура данных основной программы 6](#_Toc450247439)

[4.2 Структура данных подпрограмм 6](#_Toc450247440)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 7](#_Toc450247441)

[5.1 Схема основного алгоритма 7](#_Toc450247442)

[5.2 Схема алгоритма enterGraph(graph) 8](#_Toc450247443)

[5.3 Схема алгоритма enterVertex(result, note) 9](#_Toc450247444)

[5.4 Схема алгоритма connected(result, graph, vertexA, vertexB) 10](#_Toc450247445)

[6 Результаты расчетов и тестирование программы 11](#_Toc450247446)

[6.1 Группа тестов с найденным путём 11](#_Toc450247447)

[6.1.1 Тест 1.1 11](#_Toc450247448)

[6.1.2 Тест 1.2 12](#_Toc450247449)

[6.2 Группа тестов с ненайденным путём 13](#_Toc450247450)

[6.2.1 Тест 2.1 13](#_Toc450247451)

[6.2.1 Тест 2.2 14](#_Toc450247452)

[Приложение А 15](#_Toc450247453)

# Постановка задачи

Заданы типы:

type

TVertex = (b1, b2, b3, b4, b5, b6, b7, b8, b9, b10);

TNeighbors = set of TVertex;

TGraph = array [TVertex] of TNeightbors;

Описать логическую функцию, которая определяет есть ли в ориентированном графе путь из одной вершины в другую

# Описание алгоритмов

Таблица 1 – Описание алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры | Рекомендуемый тип |
|  | Основной алгоритм | Вызов следующих алгоритмов:  enterGraph,  enterVertex,  connected |  |  |
|  | enterGraph(graph) | Запрашивает ввод двух вершин графа, между которыми создаёт связь, если не были введены дву нуля, то вызывает себя рекурсивно | graph.  Возвращаемый параметр: graph | Процедура |
|  | enterVertex(result,note) | Выводит сообщение note, запрашивает ввод вершины result | result, note.  Возвращаемый параметр: result | Функция.  Возвращаемый параметр: result |
|  | connected(result, graph, vertexA, vertexB) | Проверяет наличие прямого пути между вершинами vertexA и vertexB, если эти вершины не связаны, то вызывает себя рекурсивно для каждой смежной с vertexA вершины | result, graph, vertexA, vertexB  Возвращаемый параметр: result | Функция. Возвращаемый параметр: Result |

# Структура типов

## Структура типов основной программы

Таблица 2 – Структура типов

|  |  |  |
| --- | --- | --- |
| Наименование типа | Рекомендуемый тип | Назначение |
| TVertex | (b1, b2, b3, b4, b5, b6, b7, b8, b9, b10) | Все вершины графа |
| TNeighbors | TNeighbors = set of TVertex | Множество смежных вершин |
| TGraph | TGraph = array [TVertex] of TNeightbors | Массив множеств соседних вершин для всех вершин графа |

# Структура данных

## Структура данных основной программы

Таблица 3 – Структура данных

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| graph | TGraph | Граф |
| vertexFrom | TVertex | Вершина начала пути |
| vertexTo | TVertex | Вершина конца пути |

## Структура данных подпрограмм

Таблица 4 – Структура данных алгоритма enterGraph(graph)

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| graph | TGraph | Заполняемый граф |
| a, b | Integer | Вершины, которые должны быть связаны |

Таблица 5– Структура данных алгоритма enterVertex(result,note)

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| result | TVertex | Введённая пользователем вершина |
| note | String | Запрос на ввод номера вершины |
| temp | Integer | Номер введённой пользователем вершины |

Таблица 6 – Структура данных алгоритма connected(result, graph, vertexA, vertexB)

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| result | Boolean | Наличие или отсутствие пути между вершинами |
| graph | TGraph | Граф, в котором производится поиск пути |
| vertexA, vertexB | TVertex | Начальная и конечная вершины пути |

# Схема алгоритма решения задачи по ГОСТ 19.701-90

## Схема основного алгоритма



Рисунок 1 – Схема основного алгоритма

## Схема алгоритма enterGraph(graph)



Рисунок 2 – Схема алгоритма enterGraph(graph)

## Схема алгоритма enterVertex(result, note)



Рисунок 3 – Схема алгоритма enterVertex(result, note)

## Схема алгоритма connected(result, graph, vertexA, vertexB)



Рисунок 4 – Схема алгоритма connected(result, graph, vertexA, vertexB)

# Результаты расчетов и тестирование программы

## Группа тестов с найденным путём

### Тест 1.1

Таблица 7 – Тест 1.1

|  |  |
| --- | --- |
| Тестовая  ситуация | Поиск пути между смежными вершинами |
| Исходные данные | Граф из 5 вершин, указанный на рисунке выше.  Поиск пути из 2 вершины в 1 |
| Ожидаемый  результат | Путь найден - TRUE |
| Полученный результат |  |

### Тест 1.2

Таблица 8 – Тест 1.2

|  |  |
| --- | --- |
| Тестовая  ситуация | Поиск пути между несмежными вершинами |
| Исходные данные | Граф из 5 вершин, указанный на рисунке выше.  Поиск пути из 2 вершины в 3 |
| Ожидаемый  результат | Путь найден - TRUE |
| Полученный результат |  |

## Группа тестов с ненайденным путём

### Тест 2.1

Таблица 9 – Тест 2.1

|  |  |
| --- | --- |
| Тестовая  ситуация | Поиск пути между несмежными вершинами  Поиск пути из вершины, смежной с другими |
| Исходные данные | Граф из 5 вершин, указанный на рисунке выше.  Поиск пути из 3 вершины в 5 |
| Ожидаемый  результат | Путь найден - FALSE |
| Полученный результат |  |

### Тест 2.2

Таблица 10 – Тест 2.2

|  |  |
| --- | --- |
| Тестовая  ситуация | Поиск пути между несмежными вершинами  Поиск пути из вершины, не смежной ни с одной другой |
| Исходные данные | Граф из 5 вершин, указанный на рисунке выше.  Поиск пути из 3 вершины в 5 |
| Ожидаемый  результат | Путь найден - FALSE |
| Полученный результат |  |

Приложение А

(обязательное)

Исходный код программы

program Lab14\_15;

{$APPTYPE CONSOLE}

uses

Windows;

type

// Перечесление всех вершин графа

TVertex = (b1, b2, b3, b4, b5, b6, b7, b8, b9, b10);

// Множество соседей вершины графа

TNeighbors = set of TVertex;

// Массив множеств соседей вершин графа

TGraph = array [TVertex] of TNeighbors;

var

graph: TGraph;

vertexFrom, vertexTo: TVertex;

{ Заправшивает ввод двух вершин графа до ввода двух нулей }

{ Создаёт в графе связь между введёнными вершинами }

procedure enterGraph(var graph: TGraph);

var

a, b: Integer;

begin

repeat

write('Введите первую вершину: ');

readln(a);

until (a >= 0) and (a <= 10);

repeat

write('Введите вторую вершину: ');

readln(b);

until (b >= 0) and (b <= 10);

writeln;

if (a <> 0) or (b <> 0) then

begin

graph[TVertex(a - 1)] := graph[TVertex(a - 1)] +

[TVertex(b - 1)];

enterGraph(graph);

end;

end;

{ Запрашивает ввод номера вершины, возвращает эту вершину }

function enterVertex(note: String):TVertex;

var

temp: Integer;

begin

repeat

write(note);

readln(temp);

until (temp <= 10) and (temp > 0);

result := TVertex(temp - 1);

end;

{ Рекурсивный алгоритм поиска в глубину }

{ Возвращает true, есть путь между введёнными вершинами }

{ Иначе возвращает false }

function connected(graph: TGraph;

vertexA, vertexB: TVertex):Boolean;

var

vert: TVertex;

begin

if vertexB in graph[vertexA] then

result := true

else

begin

result := false;

for vert := b1 to b10 do

begin

graph[vert] := graph[vert] - [vertexA];

if vert in graph[vertexA] then

result := result or connected(graph, vert, vertexB);

end;

end;

end;

begin

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

enterGraph(graph);

vertexFrom := enterVertex('Введите начальную вершину: ');

vertexTo := enterVertex('Введите конечную вершину: ');

writeln(connected(graph, vertexFrom, vertexTo));

readln;

end.