# Санкт-Петербургский государственный политехнический университет

Кафедра “Прикладная математика”

Отчет по лабораторной работе 2

“Алгоритмы и структуры данных”

Студент группы № 5030102/20002

ФИО Сатунов Буучай Юрьевич

Выполнил (дата) 29.10.2023

Оглавление

[Санкт-Петербургский государственный политехнический университет 1](#_Toc149504545)

[Постановка задачи 1](#_Toc149504546)

[Описание алгоритма 2](#_Toc149504547)

[Добавление элемента 2](#_Toc149504548)

[Печать списка 2](#_Toc149504549)

[Текст программы 2](#_Toc149504550)

[Main.cpp: 2](#_Toc149504551)

[Graphs.cpp: 3](#_Toc149504552)

[StructureOfLists.cpp: 3](#_Toc149504553)

Постановка задачи

В данной работе требуется на примере конкретной задачи реализовать преобразование матрицы смежности в список смежности. Для начала формализуем исходную задачу.

Итак, пользователь нашей программы может делать следующие запросы к ней:

* Загрузить матрицу смежности из файла;
* Преобразовать матрицу смежности в список смежности;
* Напечатать список смежности;
* Завершить программу.

# Описание алгоритма

Для описания алгоритма требуется пояснить, по каким правилам строится список смежности из матрицы смежности.

Список смежности – это способ представления графа, в котором для каждой вершины перечислены все вершины, с которыми она связана. В нашем случае, мы будем использовать пользовательский контейнер для хранения списка смежности.

Преобразование матрицы смежности в список смежности происходит следующим образом:

* Для каждой вершины i в графе, мы просматриваем i-ую строку в матрице смежности.
* Если элемент j в этой строке равен 1, это означает, что между вершинами i и j есть ребро. В этом случае, мы добавляем j в список смежности для вершины i.

# Добавление элемента

Осуществим добавление элемента в список смежности следующим образом:

* Создаем новый узел списка;
* Устанавливаем значение узла равным номеру вершины, к которой есть ребро;
* Добавляем узел в начало списка.

# Печать списка

Печать списка смежности реализуется следующим образом:

* Для каждого списка в массиве списков (каждый список соответствует одной вершине), печатаем номер вершины и все элементы списка.

# Текст программы

# Main.cpp:

#include "Graphs.cpp"

int main() {

setlocale(LC\_ALL, "Russian");

Graph graph("matrix.txt");

graph.printGraph();

return 0;

}

# Graphs.cpp:

#include <iostream>

#include <fstream>

#include "StructureOfList.cpp"

// Класс для работы с графом

class Graph {

private:

AdjacencyList\* list;

int n;

public:

Graph(const std::string& filename) : list(nullptr), n(0) {

std::ifstream file(filename);

if (!file) {

std::cout << "Не удалось открыть файл" << std::endl;

return;

}

file >> n;

list = new AdjacencyList[n];

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

int temp;

file >> temp;

if (temp == 1) {

list[i].addNode(j);

}

}

}

}

void printGraph() {

for (int i = 0; i < n; i++) {

std::cout << i << ": ";

list[i].printList();

}

}

~Graph() {

delete[] list;

}

};

# StructureOfLists.cpp:

#include <iostream>

struct Node {

int data;

Node\* next;

};

// Класс для хранения списка смежности

class AdjacencyList {

private:

Node\* head;

public:

AdjacencyList() : head(nullptr) {}

void addNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

newNode->next = head;

head = newNode;

}

void printList() {

Node\* temp = head;

while (temp != nullptr) {

std::cout << temp->data << " ";

temp = temp->next;

}

std::cout << std::endl;

}

};

Тестирование программы

Для тестирования программы был использован следующий набор данных:

5

0 1 0 0 1

1 0 1 1 0

0 1 0 1 0

0 0 1 0 1

1 0 0 1 1

Это матрица смежности для графа с пятью вершинами. Значение 1 в i-ой строке и j-ом столбце означает, что есть ребро между вершинами i и j.

После загрузки этой матрицы в программу и преобразования её в список смежности, программа вывела следующий результат:

0: 4 1

1: 3 2 0

2: 3 1

3: 4 2

4: 4 3 0

# Выводы

В целом, эта работа показала, как можно эффективно работать с графами в C++, используя матрицу смежности и список смежности. Эти знания могут быть полезны при решении более сложных задач, связанных с графами.