Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по курсу «Программирование»

на тему «Разработка программы сложной структуры методом многомодульного программирования.»

наименование программы “Реализация алгоритма Дейкстры”

Выполнил студент группы 20ВВ3:

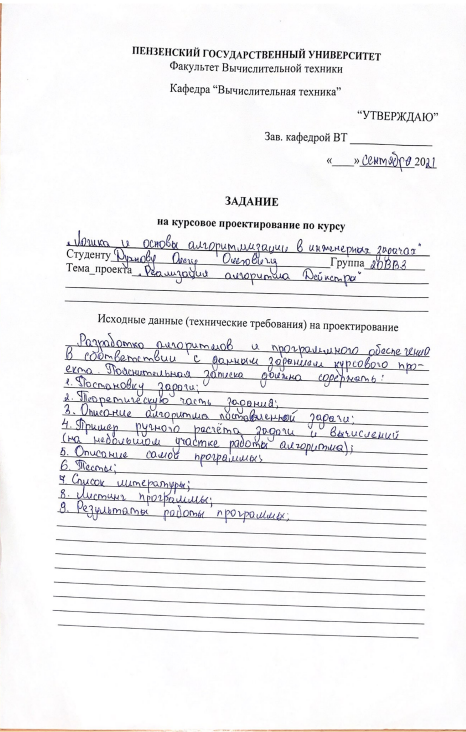
Духнов О.О.

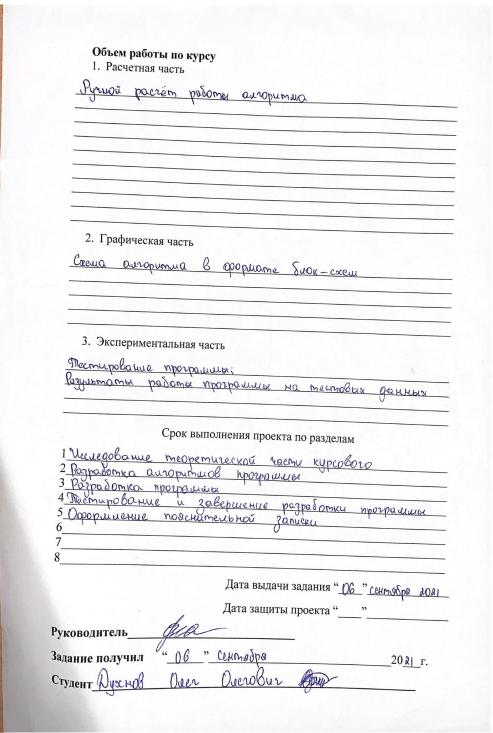
Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2021





**Введение.**

Основная задача графов – это отображать связи между разными сущностями, то есть вершинами. К примеру, это может использоваться в карте метро, на которой станции являются вершинами графа, а связывающие их перегоны – ребрами.

С помощью графов можно представить связи между пользователями социальной сети или ссылки между разными страницами одного сайта. В таких графах рёбра будут иметь направление – пользователь А подписан на пользователя Б, а не наоборот. Это ориентированные графы.

В настоящее время это очень востребованная структура данных, так как она позволяет работать с большими объёмами плохо структурированной информации. На графах, например, основываются разнообразные системы рекомендаций и ранжирования контента.

Самый популярный способ для представления графов в языках программирования – это матрицы смежности.

Основные алгоритмы на графах – это обход графа и нахождение кратчайшего пути. Есть два пути обхода вершин в графе: поиск в ширину и глубину. Существует множество алгоритмов нахождения кратчайшего пути, например, алгоритм Дейкстры.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2019, язык программирования – С++. Целью данной курсовой работы является разработка программы на языке С++, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм нахождения кратчайшего пути.

1. **Постановка задачи.**

Требуется разработать программу, которая будет находить кратчайшее расстояние от одной из вершин до всех остальных.

Исходный граф в программе должен задаваться матрицей смежности, причём при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. После обработки этих данных на экран должна выводиться матрица смежности орграфа, вид орграфа и все компоненты связности орграфа. Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно. Устройство ввода – клавиатура и мышь.

1. **Теоретическая часть задания.**

Алгоритм Дейкстры – это алгоритм, который был изобретён нидерландским учёным Э. Дейкстрой в 1959 году. Данный алгоритм предназначен для нахождения кратчайшего расстояния от одной из вершин графа до всех остальных. Работает только для графов без рёбер отрицательного веса.

Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a.

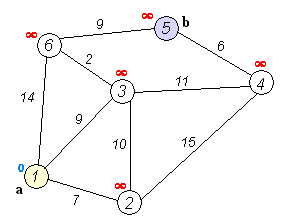
Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности.

Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как не посещённые. Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку.

Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.

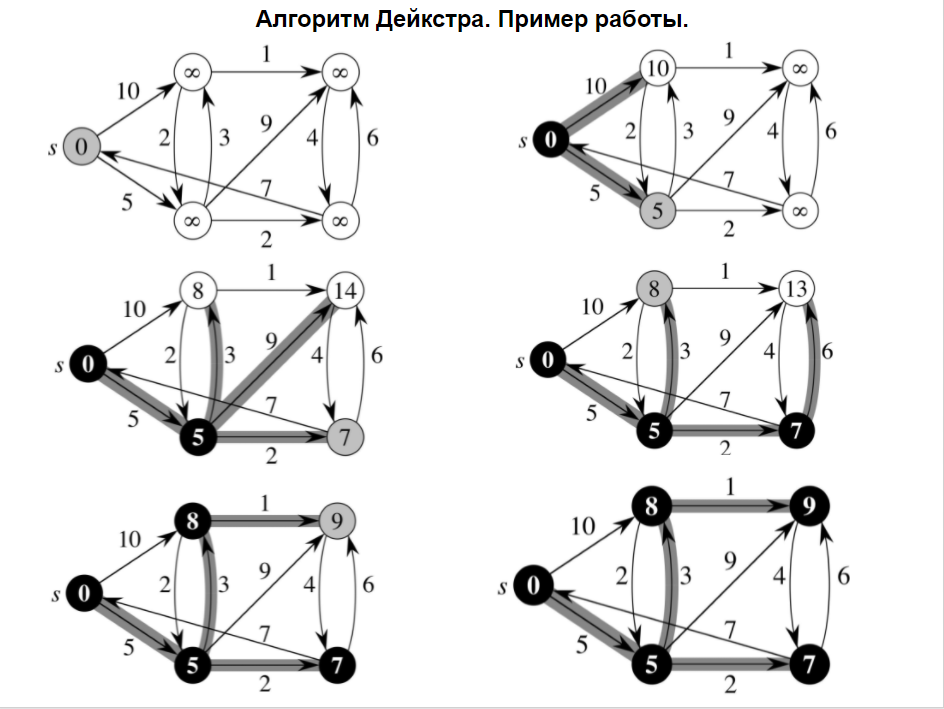
[](https://commons.wikimedia.org/wiki/File:Dijkstra_Animation.gif?uselang=ru)

1. **Описание алгоритма поставленной задачи.**

На первой итерации для s мы поставили 0, а все остальные вершины не посещали, они равны бесконечности. Дальше выбираем кратчайшее расстояние из всех смежных вершин. Со следующей итерации будем начинать с неё (с вершины, до которой самое короткое расстояние), так как есть вероятность, что найдём более короткий путь

Далее просматриваем все смежные вершины и проверяем, меньше ли будет расстояние до смежных вершин с известным путём, если мы пойдём к ним из данной вершины. Если меньше, то расстояние меняем на меньшее, иначе не меняем. В следующей итерации пойдём в вершину, к которой наименьшее расстояние.

После выполнения всех итераций, мы должны перейти в просмотренную до этого вершину и проверить, есть ли такое расстояние до смежных вершин, которое будет меньше, если есть, то меняем, иначе оставляем расстояние таким же.



1. **Пример ручного расчёта задачи и вычислений.**
2. **Описание самой программы.**

**8. Листинг программы.**

**main.cpp**

#include "Header.h"

int\*\* create\_arr(int n)

{

int\*\* Arr2 = (int\*\*)malloc(n \* sizeof(int\*));

srand(time(NULL));

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

{

Arr2[i] = (int\*)malloc(n \* sizeof(int));

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

{

if (rand() % 100 > 70)

{

Arr2[i][j] = 0;

}

else

{

Arr2[i][j] = rand() % n;

}

}

}

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

{

Arr2[i][i] = 0;

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

{

if (Arr2[i][j] > 0)

Arr2[j][i] = 0;

}

}

printf("\n");

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

printf("%2d ", Arr2[i][j]);

}

return Arr2;

}

int\*\* zapolny\_arr(int n)

{

int x, s = 1, \*\* A = (int\*\*)malloc(n \* sizeof(int\*));

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

{

A[i] = (int\*)malloc(n \* sizeof(int));

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

A[i][j] = 0;

}

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

{

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

{

system("cls");

printf("Введите элемент %d строки %d столбца: ", i + 1, j + 1);

cin >> x;

A[i][j] = x;

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

printf("%2d ", A[i][j]);

}

printf("\n");

system("pause");

}

}

return A;

}

int\*\* create\_matrix()

{

int \*\* A = (int\*\*)malloc(n \* sizeof(int\*));

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

{

A[i] = (int\*)malloc(n \* sizeof(int));

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

A[i][j] = 0;

}

return A;

}

//void Sigismund\_Dijkstra(int\*\* A, int n, int x)

//{

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

// {

// printf("\n");

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

// {

// if (A1[i][j] == SHRT\_MAX)

// printf("%2d ", 0);

// else

// printf("%2d ", A1[i][j]);

// }

// }

// int\* dist = (int\*)malloc(n \* sizeof(int));

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

// dist[i] = SHRT\_MAX;

// dist[x] = 0;

// list<int> lst;

// list<int> ls;

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

// ls.push\_back(i);

// ls.remove(x);

// while (!ls.empty())

// {

// int a, l = SHRT\_MAX;

// for (int k : ls)

// if (k != x && A[x][k] != 0 && A[x][k] < l)

// {

// l = A[x][k];

// a = k;

// }

// lst.push\_back(a);

// ls.remove(a);

// x = a;

// for (int k : ls)

// printf("%d ", k);

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

// {

// if (A[a][i] > 0)

// {

// if (dist[i] > dist[a] + A[a][i])

// {

// dist[i] = dist[a] + A[a][i];

// //printf("%d ", a);

// }

// }

// }

// }

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

// printf("%d ", dist[i]);

// printf("Алгоритм выполнен!");

// system("pause");

// menu1();

//}

void Sigismund\_Dijkstra(int\*\* A, int n, int x, int\* dist)

{

int\* num = (int\*)(malloc(n \* sizeof(int)));

int minR, minV;

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

{

num[i] = 0;

}

dist[x] = 0;

do

{

minR = SHRT\_MAX;

minV = SHRT\_MAX;

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

{

if (num[i] == 0 && dist[i] <= minR)

{

minR = dist[i];

minV = i;

}

}

if (minV != SHRT\_MAX)

{

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

{

if (A[minV][i] > 0 && dist[i] > minR + A[minV][i])

{

dist[i] = minR + A[minV][i];

}

}

num[minV] = 1;

}

} while (minV < SHRT\_MAX);

}

void create\_db()

{

char name[16];

printf("Создание БД.\nВведите имя БД: ");

scanf("%s", name);

OemToCharBuffA(name, name, strlen(name));

char folder[26] = "DB\\", format[5] = ".txt";

strcat(name, format);

strcat(folder, name);

DB = fopen(folder, "w");

fclose(DB);

system("PAUSE");

menu();

}

void create\_graf()

{

int x, SW1;

system("cls");

printf("1) Создать матрицу смежности.\n2) Ввод матрицы смежности.\n3) Назад.\nВыберите пункт: ");

do {

SW1 = \_getch();

switch (SW1)

{

default:

system("cls");

printf("Такого пункта нет!\n");

break;

case '1':

system("cls");

printf("Введите размер: ");

scanf("%d", &n);

A1 = create\_arr(n);

system("cls");

menu1();

break;

case '2':

system("cls");

printf("Введите размер: ");

scanf("%d", &n);

A1 = zapolny\_arr(n);

system("cls");

menu1();

break;

case '3':

menu1();

}

} while (SW1 != '3');

}

void menu1()

{

system("cls");

int x, \* dist = (int\*)(malloc(n \* sizeof(int))), \*\* matrix = (int\*\*)(malloc(n \* sizeof(int\*))), SW;

printf("1) Создать граф.\n2) Алгоритм Дейкстры.\n3) Просмотр результата.\n4) Сохранить результат.\n5) Загрузить из файла\n6) Назад.\nВыберите пункт: ");

do {

SW = \_getch();

switch (SW)

{

case '1':

system("cls");

create\_graf();

system("cls");

break;

case '2':

system("cls");

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

matrix[i] = (int\*)malloc(n \* sizeof(int));

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

{

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

dist[j] = SHRT\_MAX;

Sigismund\_Dijkstra(A1, n, i, dist);

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

matrix[i][j] = dist[j];

}

printf("Результат: ");

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

{

printf("\n");

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

{

if (matrix[i][j] == SHRT\_MAX)

printf("%2d ", 0);

else

printf("%2d ", matrix[i][j]);

}

}

system("pause");

break;

case '3':

system("cls");

print();

break;

case '4':

system("cls");

print\_file();

break;

case '5':

system("cls");

read\_file();

break;

case '6':

menu();

default:

system("cls");

printf("Такого пункта нет!\n");

menu1();

break;

}

} while (SW != '6');

}

void open\_db()

{

char name[16];

printf("Открытие БД.\nВведите имя БД: ");

scanf("%s", name);

OemToCharBuffA(name, name, strlen(name));

char folder[26] = "DB\\", format[5] = ".txt";

strcat(name, format);

strcat(folder, name);

strcpy(ADRESS, folder);

DB = fopen(ADRESS, "r+");

if (fopen(ADRESS, "r+") == 0)

{

printf("Такой БД не существует!\n");

system("PAUSE");

return;

}

fclose(DB);

menu1();

system("PAUSE");

}

void print\_file()

{

DB = fopen(ADRESS, "w");

fprintf(DB, "Размер массива: %d", n);

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

{

fprintf(DB, "\n");

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

if (A1[i][j] == SHRT\_MAX)

fprintf(DB, "%2d ", 0);

else

fprintf(DB, "%2d ", A1[i][j]);

}

fclose(DB);

printf("Файл успешно сохранился!");

system("pause");

}

void print()

{

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

{

if (A1[i][j] == SHRT\_MAX)

printf("%2d ", 0);

else

printf("%2d ", A1[i][j]);

}

}

system("pause");

}

void read\_file()

{

DB = fopen(ADRESS, "r");

fseek(DB, 16, SEEK\_SET);

fscanf(DB, "%d", &n);

A1 = create\_matrix();

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

{

printf("\n");

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

fscanf(DB, "%d", &A1[i][j]);

}

fclose(DB);

printf("Файл успешно загружен!");

\_getch();

}

void menu()

{

system("cls");

printf("1) Создать базу данных.\n2) Открыть базу данных.\n3) Выход. \nВыберите пункт: ");

int SW2;

do {

SW2 = \_getch();

switch (SW2)

{

case '1':

system("cls");

create\_db();

system("cls");

menu();

break;

case '2':

system("cls");

open\_db();

system("cls");

break;

case '3':

exit(0);

default:

system("cls");

printf("Такого пункта нет!\n");

break;

}

} while (SW2 != '3');

}

int main()

{

setlocale(LC\_ALL, "Ru");

system("cls");

menu();

}

**Header.h**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <stdbool.h>

#include <conio.h>

#include <time.h>

#include <list>

#include <string.h>

#include <windows.h>

using namespace std;

FILE\* DB;

int\*\* A1, n;

char ADRESS[32];

void menu1();

void menu();

void print\_file();

int\*\* create\_arr(int x);

int\*\* zapolny\_arr(int n);

void print();

void read\_file();