Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Основы алгоритмизации и программирования

Отчёт

по учебной практике(ознакомительной)

на тему:

Программа для реализации алгоритма Дейкстры (нахождения кратчайших расстояний)

Студент Д.И.Ковальчук

Руководитель А.М.Ковальчук

Минск 2022

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ ...............................................................................................................

1. ВХОДНЫЕ ДАННЫЕ...........................................................................................

2. ВЫХОДНЫЕ ДАННЫЕ. .....................................................................................

3. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ..................................................

3.1. БЛОК-СХЕМА ГОЛОВНОГО МОДУЛЯ........................................................

3.2. АЛГОРИТМ ПО ШАГАМ ФУНКЦИИ shortestways()..............................................................................................................

3.3. АЛГОРИТМ ПО ШАГАМ ФУНКЦИИ exactway()……….......................................................................................................

4. КОД ПРОГРАММЫ..............................................................................................

4.1 main.cpp................................................................................................................

4.2 funcs.cpp………....................................................................................................

4.3 hhhpract.h……......................................................................................................

5. РЕЗУЛЬТАТЫ……...............................................................................................

6.ЗАКЛЮЧЕНИЕ .....................................................................................................

7. ПРИМЕНЕНИЕ….................................................................................................

8.СПИСОК ЛИТЕРАТУРЫ .....................................................................................

**ПРИЛОЖЕНИЕ А**...................................................................................................

**ПРИЛОЖЕНИЕ Б**...................................................................................................

**ПРИЛОЖЕНИЕ В**...................................................................................................

**ВВЕДЕНИЕ**

Язык С часто называют языком среднего уровня. Это означает, что С объединяет элементы языков высокого уровня с функциональностью Ассемблера. Языки высокого уровня поддерживают концепцию типов данных. Тип данных определяет набор значений, которые переменная может хранить, и набор операций, которые могут выполняться над переменными. Наряду с тем, что в языке С представлены все основные типы данных, он не так жестко типизирован, как языки Паскаль или Ада. Язык С позволяет осуществлять большинство преобразований типов. Контроль за выполнением этих преобразований, а также проверка некоторых ошибок (например, выход за границы массива) возлагается на программиста.

Реализованная в С возможность напрямую манипулировать битами, байтами, словами и указателями необходима для программирования на системном уровне.

Язык С считается структурированным языком. Отличительной чертой структурированного языка является разделение кода и данных. Одним из способов решения этой проблемы является использование подпрограмм (функций), широко использующих локальные переменные. Необходимо отметить, что излишнее использование глобальных переменных может приводить к фатальным ошибкам.

Как и ряд других структурированных языков, С поддерживает ряд операторов цикла, условных операторов и операторов ветвления. Наряду с этим нежелательно использование оператора goto.

Язык С содержит стандартные библиотеки, предоставляющие функции, выполняющие наиболее типичные задачи. Эти библиотеки легко могут быть подключены, а также дополнены.

Язык С позволяет разбивать программу на части и выполнять их раздельную компиляцию. Откомпилированные таким образом файлы объединяются для создания полного объектного кода. Преимущество раздельной компиляции в том, что при изменении одного файла не требуется перекомпиляции всей программы.

Алгоритм Дейкстры

Алгоритм Дейкстры — это метод, который находит кратчайший путь от одной вершины графа к другой. Граф — структура из точек-вершин, соединенных ребрами-отрезками. Его можно представить как схему дорог или как компьютерную сеть. Ребра — это связи, по ним можно двигаться от одной вершины к другой.

Алгоритм Дейкстры работает для графов, у которых нет ребер с отрицательным весом, т.е. таких, при прохождении через которые длина пути как бы уменьшается.

В отличие от похожих методов, алгоритм Дейкстры ищет оптимальный маршрут от одной заданной вершины ко всем остальным. Попутно он высчитывает длину пути — суммарный вес ребер, по которым проходит при этом маршруте.

Алгоритм Дейкстры пошаговый. Сначала выбирается точка, от которой будут отсчитываться пути. Затем алгоритм поочередно ищет самые короткие маршруты из исходной точки в другие. Вершины, где он уже побывал, отмечает посещенными. Алгоритм использует посещенные вершины, когда рассчитывает пути для непосещенных.

**ВХОДНЫЕ ДАННЫЕ**

1) Количество вершин и матрица смежности графа, введённые пользователем

2) Количество вершин и матрица смежности, прочитанные из файла

3) Начальная и конечная вершины графа, между которыми нужно найти расстояние и кратчайший путь

Количество вершин и матрица смежности, в зависимости от выбора пользователя, могут быть введены в консоль, либо прочитаны из текстового файла

int size – количество вершин графа;

int\*\* arr – матрица смежности графа;

int k1 – начальная вершина, от которой будут считаться расстояния и пути

int k11- конечная вершина, до которой будет найдено кратчайшее расстояние и кратчайший путь от начальной вершины

crear.txt – текстовый файл, хранящий количество вершин и матрицу смежности

**ВЫХОДНЫЕ ДАННЫЕ**

1. Кратчайшие расстояния от начальной вершины до каждой вершины графа.

2. Кратчайшие расстояния от начальной до конечной вершины графа отдельно от п.1

3. Кратчайший путь от начальной до конечной вершины графа.

Результат, в зависимости от выбора пользователя, может быть дозаписан в текстовый файл.

int\* d – массив целых чисел, хранящий кратчайшие расстояние от начальной вершины до каждой из вершин

int\* ver – массив целых чисел, хранящий вершины, которые составляют путь от начальной до конечной вершины

**РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

1. **БЛОК-СХЕМА ГОЛОВНОГО МОДУЛЯ:**

Блок-схема головного модуля представлен в **ПРИЛОЖЕНИИ А.**

1. **АЛГОРИТМ ПО ШАГАМ ФУНКЦИИshortestways():**

Алгоритм по шагам представлен в **ПРИЛОЖЕНИИ Б.**

1. **АЛГОРИТМ ПО ШАГАМ ФУНКЦИИ exactway():**

Алгоритм по шагам представлен в **ПРИЛОЖЕНИИ В.**

**КОД ПРОГРАММЫ**

* 1. Код main.cpp

#include "hhhpract.h"

int main()

{

rewind(stdin);

setlocale(LC\_ALL, "RUSSIAN"); //русификация консоли

system("cls");

int repeat; //переманная для зацикливания программы

do{

int choose;

printf("Выберите метод ввода:\n1 - С клавиатуры\n0 - Считать из файла\n");

scanf\_s("%d", &choose);

int size; //количество вершин графа

if (choose == 1)

{

size = entersize(); //ввод количества вершин с клавиатуры

}

else

{

size = readsize(); //чтение количества вершин из файла

}

system("cls"); //очистка экрана

int\*\* arr = memory(size); //выделение памяти под матрицу

//смежности

int\* d = memory1(size); // выделение памяти под массив,

//хранящий кратчайшие расстояния

if (choose == 1)

{

vvod(arr, size); //ввод матрицы смежности с клавиатуры

}

else

{

readgraph(arr,size); //чтение матрицы смежности из файла

}

show(arr, size); //вывод матрицы смежности

int a; //переменная для повторной работы с графом

do

{

rewind(stdin);

int k1, k11;

printf("Введите начальную вершину: ");

scanf\_s("%d", &k1);

printf("Введите конечную вершину: ");

scanf\_s("%d", &k11);

shortestways(arr, d, size, k1); //нахождение кратчайших расстояний

int\* ver = memory1(size); //выделение памяти под массив,

//хранящий кратчайший путь

int k; //кол-во вершин в кратчайшем путе

exactway(ver, d, arr, size, k1, k11, k); //нахождение кратчайшего пути

printf("Занести результат в файл? 1 - да");

int pick;

scanf\_s("%d", &pick);

output1(size, d, ver, pick, k1, k11, k); //вывод результата

printf("\nХотите повторить для данной матрицы? Да - 1\n");

scanf\_s("%d", &a);

} while (a == 1);

printf("\nХотите продолжить программу? Да - 1, Нет - 0\n");

while (!scanf\_s("%d", &repeat) || repeat != 1 && repeat != 0)

{

printf("Enter 1 or 0\n");

rewind(stdin);

}

if (repeat == 0) return 0;

} while (repeat == 1);

}

2.Код funcs.cpp

#include "hhhpract.h"

//выделение памяти под матрицу

int\*\* memory(int size)

{

int\*\* arr = (int\*\*)calloc(size, sizeof(int\*));

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

arr[i] = (int\*)calloc(size, sizeof(int));

return(arr);

}

//выделение памяти под одномерный массив

int\* memory1(int size)

{

int\* arr = (int\*)calloc(size, sizeof(int));

return(arr);

}

//функция для ввода матрицы с клавиатуры

void vvod(int\*\* arr, int SIZE)

{

int temp;

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

{

arr[i][i] = 0;

for (int j = i + 1; j < SIZE; j++)

{

printf("Введите расстояние %d - %d: ", i + 1, j + 1);

while (!scanf\_s("%d", &temp) || temp < 0)

{

printf("Введите целое положительное число!!!\n");

rewind(stdin);

}

arr[i][j] = temp;

arr[j][i] = temp;

}

}

rewind(stdin);

}

//функция вывода матрицы

void show(int\*\* arr, int SIZE)

{

printf("Матрица:\n");

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

{

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

{

printf("%5d ", arr[i][j]);

}

printf("\n");

}

}

//функция ввода числа вершин

int entersize()

{

printf("Введите число вершин графа: ");

int size = 0;

while (!scanf\_s("%d", &size) || size < 2)

{

printf("Ошибка ввода!!!\n");

rewind(stdin);

}

return size;

}

//функция чтения числа вершин из файла

int readsize()

{

FILE\* f = fopen("creat.txt", "r"); //открытие файла

int size = 0;

fscanf(f, "%d", &size); // считывание числа

fclose(f); // закрытие файла

return size;

}

//функция чтения матрицы из файла

void readgraph(int\*\* arr, int size)

{

FILE\* f = fopen("creat.txt", "r"); //открытие файла

int x;

fscanf(f, "%d", &x);

for (int i = 0; i < size; ++i) //считывание матрицы

{

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

{

fscanf(f, "%d", &arr[i][j]);

}

putchar('\n');

}

fclose(f); //закрытие файла

}

//функция нахождения кратчайших расстояний

void shortestways(int\*\* arr, int\* d, int size, int k1)

{

int\* v = memory1(size); //посещенные вершины

int minindex, min, temp;

int begin\_index = k1 - 1;

for (int i = 0; i < size; i++) //инициализация вершин и

//расстояний

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

do //шаг алгоритма

{

minindex = 10000;

min = 10000;

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

{

if ((v[i] == 1) && (d[i] < min)) //если вершину ещё не

//обошли и вес меньше min

{

min = d[i]; //переприсывание значения

minindex = i;

}

}

if (minindex != 10000)

{

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

{

if (arr[minindex][i] > 0)

{ //добавление найденного минимального

temp = min + arr[minindex][i]; //веса к текущему весу вершины

if (temp < d[i]) //сравн. с текущим мин.весом вершины

{

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < 10000);

}

//функция нахождения кратчайшего пути

void exactway(int\* ver,int\* d,int \*\* arr, int size, int k1, int k11, int k)

{

k = 1; //индекс предыдущей вершины

int begin\_index = k1 - 1;

int end = k11 - 1;

ver[0] = end + 1; // начальный элемент - конечная вершина

int weight = d[end]; // вес конечной вершины

while (end != begin\_index) // пока не дошли до начальной вершины

{

for (int i = 0; i < size; i++) // просматриваем все вершины

if (arr[i][end] != 0) // если связь есть

{

int temp = weight - arr[i][end]; // определяем вес пути из предыдущей вершины

if (temp == d[i]) // если вес совпал с рассчитанным

{ // значит из этой вершины и был переход

weight = temp; // сохраняем новый вес

end = i; // сохраняем предыдущую вершину

ver[k] = i + 1; // и записываем ее в массив

k++;

}

}

}

}

//функция вывода результата

void output1(int size, int\* d, int\* ver, int pick,int k1, int k11, int k)

{

FILE\* f = fopen("creat.txt", "a+");

printf("\nКратчайшие расстояния до вершин: \n");

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

{

printf("Расстояние до вершины %d: %d \n", i + 1, d[i]);

if (pick == 1)

{

fprintf(f,"Расстояние до вершины %d: %d \n", i + 1, d[i]);

}

}

printf("\nКратчайшие расстояния от %d-ой до %d-ой вершины: ", k1, k11);

printf("%d ", d[k11 - 1]);

if (pick == 1)

{

fprintf(f,"\nКратчайшие расстояния от %d-ой до %d-ой вершины: ", k1, k11);

fprintf(f,"%d ", d[k11 - 1]);

}

// Вывод пути (начальная вершина оказалась в конце массива из k элементов)

printf("\nВывод кратчайшего пути: ");

for (int i = k - 1; i >= 0; i--)

{

if (ver[l] != 0)

{

printf("%3d ", ver[i]);

if (pick == 1)

{

fprintf(f,"%3d ", ver[i]);

}

}

}

fclose(f);

}

3. Код hhhpract.h

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <locale.h>

#include <windows.h>

//выделение памяти под матрицу

int\*\* memory(int size);

//выделение памяти под одномерный массив

int\* memory1(int size);

//функция ввода матрицы

void vvod(int\*\* arr, int size);

//функция вывода матрицы

void show(int\*\* arr, int size);

//функция ввода количества вершин с клавиатуры

int entersize();

//функция чтения количества вершин из файла

int readsize();

//функция чтения матрицы из файла

void readgraph(int\*\* arr, int size);

//функция нахождения кратчайших расстояний

void shortestways(int\*\* arr, int\* d, int size, int k1);

//функция вывода результата

void output1(int size, int\* d, int\* ver, int pick, int k1, int k11, int k);

//функция нахождения кратчайшего пути

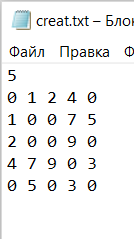
void exactway(int\* ver, int\* d, int\*\* arr, int size, int k1, int k11, int k);

**РЕЗУЛЬТАТЫ**

**ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАЧИ ПРОГРАММОЙ**

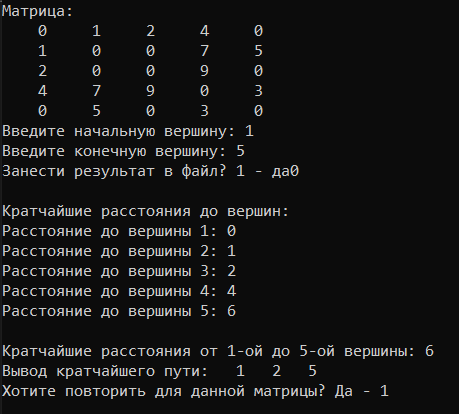
Количество вершин и матрица смежности и приведены на рисунке 1 (первая строка файла – количество вершин, последующие – матрица смежности).

Рис.1 – пример файла creat.txt



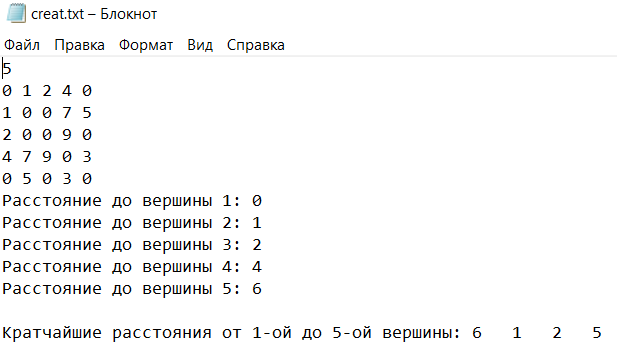
На рисунке 2 представлен пример работы программы по исходным данным.

Рис.2 – пример работы программы.



После завершения работы программы результаты могут быть дозаписаны в тот же текстовый файл (рис. 3).

Рис. 3– пример дозаписи файла



**ЗАКЛЮЧЕНИЕ**

**Условия и ограничения для запуска программы**:

* Все вводимые значения ограничиваются диапазоном типа int (32767)
* Программа может быть скомпилирована с помощью любого Windows-совместимого компилятора и запущена на любом компьютере под управлением ОС Microsoft Windows.

**ПРИМЕНЕНИЕ**

Основная задача алгоритма Дейкстры — поиск кратчайшего пути по схеме, где множество точек соединено между собой отрезками. В виде такой схемы можно представить многие объекты реального мира, поэтому практических примеров использования алгоритма много:

* автоматическое построение маршрута на онлайн-карте;
* поиск системой бронирования наиболее быстрых или дешевых билетов, в том числе с возможными пересадками;
* моделирование движения робота, который перемещается по местности;
* разработка поведения неигровых персонажей, создание игрового ИИ в разработке игр;
* автоматическая обработка транспортных потоков;
* маршрутизация движения данных в компьютерной сети;
* расчет движения тока по электрическим цепям.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Луцик Ю.А., Ковальчук А.М., Сасин Е.А. – Учебное пособие по курсу «Основы алгоритмизации и программирования: язык С». – Минск: БГУИР, 2015 г.

2. Луцик Ю.А., Ковальчук А.М., Лукьянова И.В., Бушкевич А.В. – Лабораторный практикум по курсу «Основы алгоритмизации и программирования» (в 2-х частях). – Минск: БГУИР, 2008 г.

**ПРИЛОЖЕНИЕ А**

**ПРИЛОЖЕНИЕ Б**

**ПРИЛОЖЕНИЕ В**

Алгоритм функции нахождения кратчайших расстояний

void shortestways(int\*\* arr, int\* d, int size, int k1)

1) Начало

2) Входные данные:

int\*\* arr – матрица смежности;

int\* d – массив, который будет хранить кратчайшие расстояния до вершин;

int size – количество вершин графа;

int k1 – начальная вершина;

int \*v – массив, хранящий посещенные вершины;

int begin\_index=k1-1;

3) Выделение памяти под массив v на размер size;

4) Начало цикла с параметром i=0, пока i не равен size, с увеличением i на 1

5) Инициализация i-го элемента массива d как 10000, а очередного элемента массива v 1.

d[i] = 10000;

v[i] = 1;

6) Конец цикла

7) Элемент массива, которому соответствует начальная вершина, инициализируется 0

d[begin\_index] = 0;

8) Начало цикла с постпроверкой с параметром minindex, пока параметр меньше 10000

9) Инициализация массива как можно большими числами, согласно

minindex = 10000;

min = 10000;

10) Начало цикла с параметром i=0, пока i не равен size, с увеличением i на 1

11) Если вершину ещё не обошли и вес меньше min, переприсваивание значений

min = d[i];

minindex = i, иначе продолжение цикла.

12) Конец цикла

13) Если minindex не равен 10000, переход на п.14, иначе переход на п.

14) Начало цикла с параметром i=0, пока i не равен size, с увеличением i на 1

15) Если элемент матрицы смежности arr с индексами minindex, i больше нуля, переход на п.16, иначе продолжение цикла

16) Добавляем найденный минимальный вес к текущему весу вершины

temp = min + arr[minindex][i];

17) Сравнение с текущим весом вершины d[i], если он меньше, замена минимального веса вершины на текущий

d[i] = temp;

18) Конец цикла

19) Конец цикла с постпроверкой.

20) Конец

Алгоритм функции нахождения кратчайшего пути

void exactway(int\* ver, int\* d, int \*\* arr, int size, int k1, int k11)

1) Начало

2) Входные данные

int\*\* arr – матрица смежности;

int\* d – массив, хранит кратчайшие расстояния до вершин;

int\* ver – массив, который будет хранить путь

int size – количество вершин графа;

int k1 – начальная вершина;

int k11 – конечная вершина;

int begin\_index=k1-1 – начальный индекс массива

int end = k11 - 1; - конечный индекс массива

3)

Начальный элемент - конечная вершина ver[0] = end + 1;

Индекс предыдущей вершины int k = 1

Вес конечной вершины int weight = d[end]

4) Начало цикла с параметром end, пока не дошли до начальной вершины

5) Начало цикла с параметром i=0, пока i не равен size, с увеличением i на 1

6) Если между вершинами есть связь, переход на п.7, иначе продолжение цикла

7) Определяем вес пути из предыдущей вершины

int temp = weight - arr[i][end];

8) Если вес совпал с рассчитанным, переход на п.9

9) Сохранение нового веса weight = temp, сохранение предыдущей вершины end = i, запись ее в новый массив ver[k] = i + 1, с увеличением k на 1

10) Конец цикла

11) Конец цикла

12) Конец