Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №8  
по дисциплине: «Обход графа в ширину»

Выполнили студенты группы 22ВВВ2:  
Ипполитов И. Д.

Сергунов М. Р.

Приняли:  
Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Название**

Определение характеристик графов

**Цель работы**

Разобраться в графах и матрицах. Научиться определять характеристики графов.

**Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру обхода в ширину, реализованную в соответствии с приведенным выше описанием. При  реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

**3.**\* Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

**Задание 2\***

1. Для матричной формы представления графов реализуйте алгоритм обхода в ширину с использованием очереди, построенной на основе структуры данных «список», самостоятельно созданной в лабораторной работе № 3.

Оцените время работы двух реализаций алгоритмов обхода в ширину (использующего стандартный класс **queue** и использующего очередь, реализованную самостоятельно) для графов разных порядков.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <windows.h>

#include <stdbool.h>

//#define NO\_PRINT

typedef struct node {

int x;

struct node\* next; // ссылка на следующий элемент

} node;

void printMatrix(int\*\* matrix, int size) {

printf("# ");

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

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

}

printf("\n");

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

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

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

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

}

printf("\n");

}

}

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setbuf(stdout, NULL);

//Часть 1

int size = 0;

printf("Enter a count of headers: ");

scanf(" %d", &size);

srand(time(NULL)); // Инициализация генератора случайных чисел

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

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

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

}

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

for (int j = i + 1; j < size; j++) {

int random = rand() % 2; // Генерация случайного числа 0 или 1

matrix[i][j] = random;

matrix[j][i] = random; // Заполнение элементов и их симметричных пар

}

}

// int matrix[6][6] = {

// {0, 1, 1, 0, 0, 1},

// {1, 0, 0, 1, 0, 0},

// {1, 0, 0, 0, 1, 0},

// {0, 1, 0, 0, 0, 0},

// {0, 0, 1, 0, 0, 0},

// {1, 0, 0, 0, 0, 0},

// };

#ifndef NO\_PRINT

printMatrix(matrix, size);

#endif

// 2

int\* num = (int\*)calloc(size, sizeof(int)); // 1.1

int i = 1;

#ifndef NO\_PRINT

printf("enter start vertex number:\n");

scanf("%d", &i);

printf("----\n");

#endif

i--;

setvbuf(stdout, NULL, \_IOFBF, 16384);

clock\_t time\_start = clock();

while (true) { // 1.2.

bool done = true;

for (; i < size; i++) {

if (num[i] == 0) {

done = false;

break;

}

}

if (done)

break;

// 1.3.

node\* q\_first = calloc(1, sizeof(struct node)); // 2.1.

node\* q\_last = q\_first;

q\_first->x = i; // 2.2.

q\_first->next = NULL;

num[i] = true; // 2.3.

while (q\_first != NULL) { // 2.4.

node\* f = q\_first; // 2.5.

q\_first = f->next; // 2.6.

if (q\_first == NULL)

q\_last = NULL;

#ifndef NO\_PRINT

printf("%d\n", f->x + 1); // 2.7.

#endif

free(f);

for (int i2 = 1; i2 < size; i2++) { // 2.8.

if (matrix[i][i2] == 1 && num[i2] == false) { // 2.9.

if (q\_last) {

q\_last->next = calloc(1, sizeof(struct node)); // 2.11

q\_last = q\_last->next;

}

else {

q\_first = calloc(1, sizeof(struct node)); // 2.11

q\_last = q\_first;

}

q\_last->x = i2;

q\_last->next = NULL;

num[i2] = true; // 2.12.

}

}

}

}

clock\_t time\_end = clock();

fflush(stdout);

setbuf(stdout, NULL);

double diff = time\_end - time\_start;

printf("time elapsed: %fms", diff);

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

free(matrix[i]);

free(matrix);

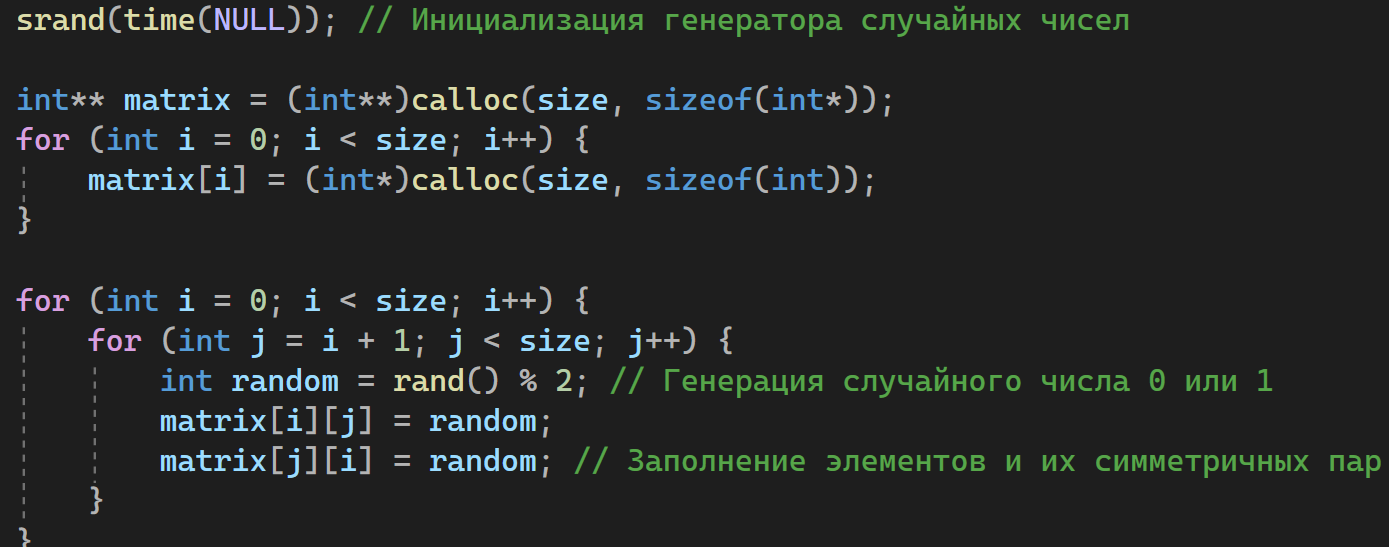
free(num);

return 0;

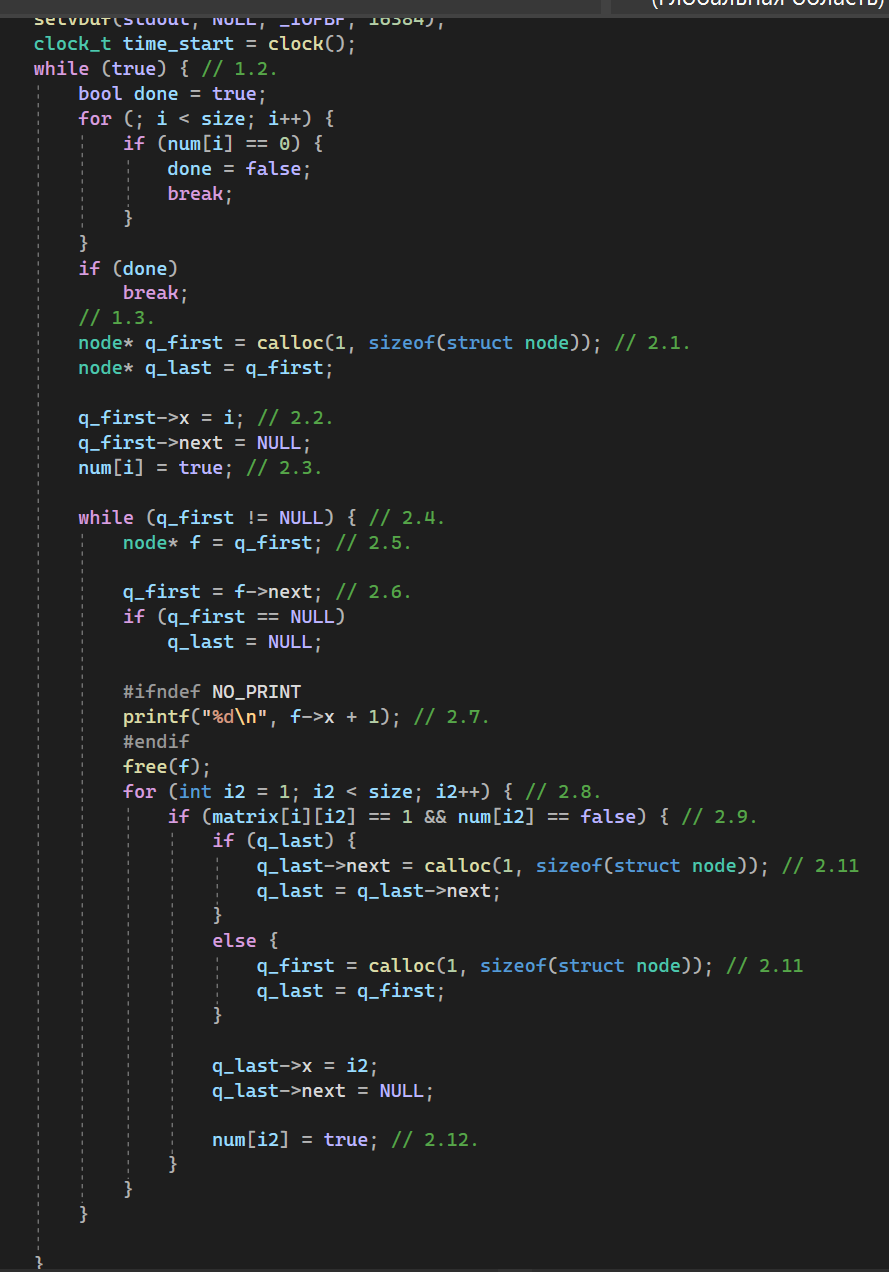
}

**Задания**

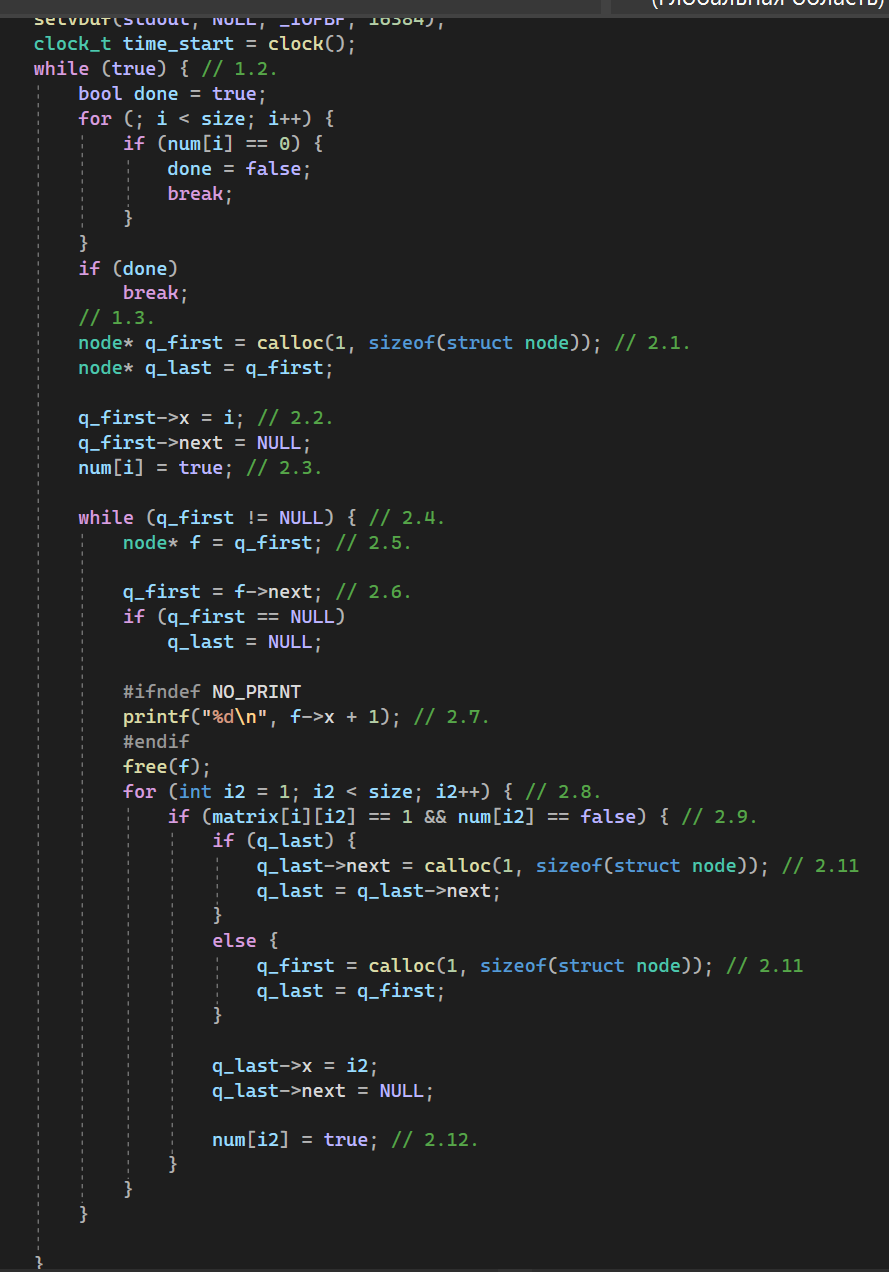
* 1. Матрица смежности



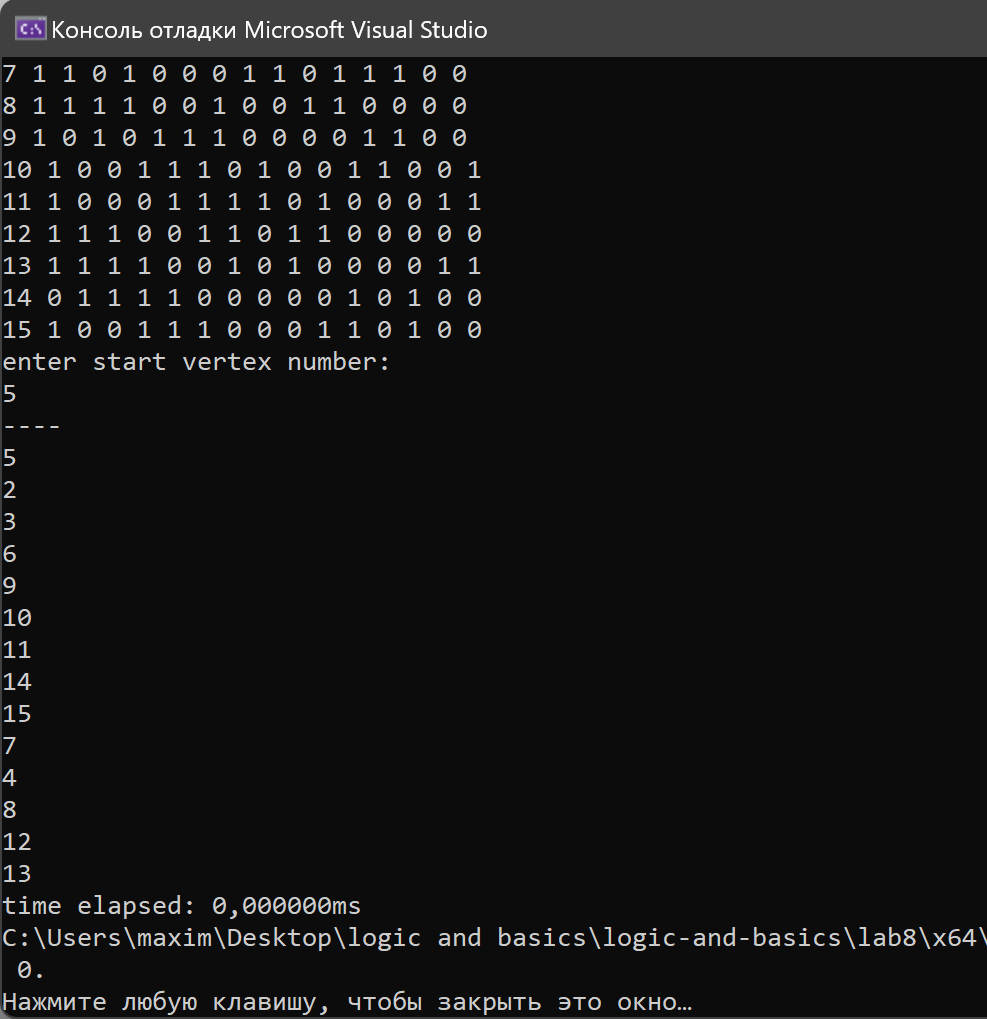
* 1. Процедура обхода в ширину для графа



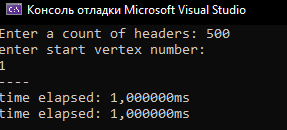
2.1 Алгоритм обхода в ширину с использованием очереди



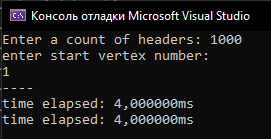
**Результат работы программы**

****

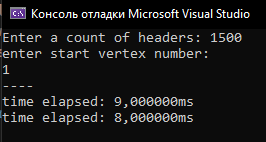
**Результат при 500 вершинах**

****

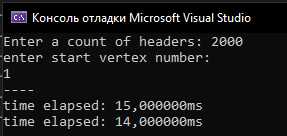
**Результат при 1000 вершинах**

****

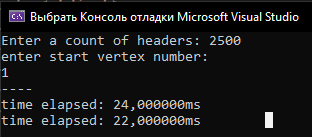
**Результат при 1500 вершинах**

****

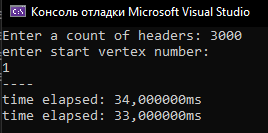
**Результат при 2000 вершинах**

****

**Результат при 2500 вершинах**

****

**Результат при 3000 вершинах**

****

**Вывод**

Разобрались с графами и матрицами, научились определять характеристики графов, создавать графы, строить алгоритмы обхода графов в ширину.

С увеличением количества вершин реализация очереди через стандартную библиотеку c++ оказывается быстрее.