КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. И. Раззакова

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Программное обеспечение компьютерных систем»

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

Методические указания к выполнению курсовой работы

Бишкек – 2009

РАССМОТРЕНЫ
На заседании кафедры
«Программное обеспечение
компьютерных систем»
Прот. № 12 от 27.05.2009г.

ОДОБРЕНЫ
Методическим советом ФИТ

Прот. № 11от 27.05.2009г.

УДК 004.432
Составитель МАКИЕВА З.Д.

Программирование на языке высокого уровня: методические указания к выполнению курсовой работы / Кырг. гос. техн. ун-т; Сост. З.Д.Макиева Бишкек, 2009.- 16 с.

Приведены требования к выполнению и оформлению курсовой работы по дисциплине “Программирование на языке высокого уровня”, а также пример выполненной курсовой работы.

Предназначены для студентов специальности «Программное обеспечение вычислительной техники и автоматизированных систем» всех форм обучения.

Библиогр. 5, названий, Прил. 1

Рецензент к.т.н. Баракова Ж.Т.

ОБЩИЕ ПОЛОЖЕНИЯ

Цель методических указаний - оказать помощь студентам в выполнении и оформлении курсовой работы по курсу “Программирование на языке высокого уровня”.

Целью курсовой работы является закрепление и углубление знаний, полученных студентами в процессе изучения дисциплины. В результате выполнения курсовой работы студент должен показать умение самостоятельно решить сложную задачу и реализовать ее на С++, применив знания по следующим разделам:

- модульное программирование (функции);
- динамическое выделение памяти;
- файловый и форматированный ввод/вывод;
- абстрактные типы данных (массивы и структуры).

При выполнении курсовой работы развиваются собственно навыки программирования и отладки программ, а также проверяется умение оформления документации на свои разработки в соответствии со стандартами и навыки публичных выступлений при защите курсовой работы.

Содержание курсовой работы

1. Условие задачи в том виде, который приведен в задании, выданном преподавателем.
2. Постановка задачи, которая включает подробное описание задачи: приводятся имена исходных данных (массивов, переменных), имена промежуточных и результирующих данных, имена файлов. Если необходимо по условию описать математическую модель задачи, т.е. определить формулы, которые будут использованы в вычислениях.
3. Графическое представление алгоритма решения в виде блок-схем (сначала блок-схема для главной функции main, а затем для каждой функции, созданной студентом).
4. Словесный (пошаговый) алгоритм решения.
5. Программа на языке С++.
6. Тестовые примеры: файлы с исходными данными и файлы с результатами.
7. Список литературы.

Задача должна быть реализована в виде функций, если возможно с использованием структур.

Исходные данные считываются из файла (файлов). Результат записывается в файл с обязательным применением форматированного вывода. При работе с файлами необходима проверка правильности их открытия.

Если в задаче речь идет о массивах, они должны быть динамическими.

Программа при выполнении должна сопровождаться диалоговыми сообщениями, т.е. пояснять свои действия (иметь понятный для пользователя

интерфейс). Текст программы на языке С++ должен комментироваться. Курсовая работа имеет не менее 15 страниц печатного текста.

Курсовой проект может включать следующие темы:

- обработка одномерных и двумерных массивов;
- обработка символьных массивов;
- работа с абстрактными типами данных структурами.

Критерии оценки курсовой работы:

1. Степень самостоятельности выполнения. Автор работы должен быть в состоянии объяснить каждую строчку в программе.

2. Оформление работы. Люди, претендующие на высшее образование, не должны делать ошибок в государственном и официальном языках. Работы с орфографическими и синтаксическими ошибками к защите не допускаются.

3. Все графические материалы (блок-схемы) должны быть выполнены в соответствии с ГОСТами.

4. Правильность работы программы при различных входных данных. Тесты должны быть проведены для критических и исключительных условий.

5. Оптимальность выбранного алгоритма с точки зрения экономии памяти и быстродействия.

Записка оформляется на русском или английском языках на листах формата А4, поля слева 30 мм, справа – 15мм, сверху и снизу – 20 мм, шрифт Times 14 пт. через 1 интервал, абзацный отступ 1,25см. Абзацы выравниваются по ширине, заголовки – по центру. Точки после заголовков не ставятся. Образец отчета по выполнению курсовой работы приведен в приложении 1.

При выполнении работы рекомендуется придерживаться следующей последовательности действий:

1. Анализ условия задачи и выработка подхода к ее решению.
2. Разработка, обоснование и описание алгоритма.
3. Выбор и обоснование тестов.
4. Выбор представления данных.
5. Программирование.
6. Подготовка программы к выполнению на ПК и ее отладка.
7. Тестирование программы и анализ результатов.
8. Оформление отчета.

Однако нужно понимать, что это разбиение условно, так как фактически все виды работ тесно переплетены и составляют единый процес решения задачи.

Сначала студент должен понять поставленную перед ним задачу, ознакомиться с соответствующими разделами учебной и рекомендованной литературы. На данном этапе студент должен ясно себе представить цели решаемой задачи и внимательно проанализировать требования, предъявляемые к его разработке.

При написании программы нужно следовать принципам структурного программирования, программа должна быть наглядной, используемые идентификаторы – содержательно осмысленными. Основные циклы, логически выделенные части программы, необходимо сопровождать комментариями и оформлять в виде подпрограмм.

Обрабатываемые данные делятся на входные, промежуточные и выходные. Специфика входных и выходных данных состоит в том, что с ними имеет дело не только алгоритм, но и пользователь программы. Поэтому различают две формы представления – внешнюю и внутреннюю. Основное требование к внешнему представлению состоит в его максимальном удобстве, понятности и естественности для пользователя, чтобы он мог достаточно легко подготовить данные для ввода и оценить результат по выходным данным без дополнительных преобразований. Выбор внутреннего представления определяется главным образом требованием эффективности алгоритма, который нужно построить для решения задачи, и может оказать существенное влияние как на объем используемой памяти, так и на время работы алгоритма.

С разработкой алгоритма и программы тесно связано и построение набора тестов, на которых работа программы будет проверятся во время отладки. При совместной разработке алгоритма и тестов легче добиться полноты проверки его правильности.

Каждый тест представляет собой набор входных данных и результатов, которые должны быть получены программой. Эти результаты получаются при решения задачи “вручную” и способствуют более точному пониманию условия задачи и выбранного алгоритма ее решения.

Набор тестов должен по возможности быть полным, т.е. проверять все части программы и особенно поведение программы “на границах” (при максимальных и минимальных значениях, самых простых и самых сложных вариантах и т.д.). Часть тестов может быть локальной, предназначенных для проверки правильности некоторых частей алгоритма, часть – комплексной, для проверки правильности решения всей задачи.

При разработке алгоритма необходимо использовать метод пошаговой детализации, т. е. разрабатывать алгоритм “сверху вниз”. На каждом этапе принимается небольшое число решений, приводящих к постепенной детализации (уточнению) управляющей и информационной структуры алгоритма.

Этот подход позволяет разбить алгоритм на части, каждая из которых решает самостоятельную подзадачу и реализуется в виде отдельной процедуры или функции. Связи по управлению ними осуществляются посредством их вызовов, а передача информации – через параметры и

глобальные переменные. При разработке алгоритма следует обратить внимание на доказательства его правильной работа (обоснование по “построению”, математические доказательства).

Шаги алгоритма должны быть представлены в виде блок-схемы и в виде словесного описания, максимально приближенного к конструкциям языка программирования в терминах тех объектов и отношений между ними, о которых идет речь в формулировке задачи.

Литература

1. Дейтел Харви, Дейтел Пол. Как программировать на C++. Пер. с англ. - Москва: ЗАО "Издательство БИНОМ", 1998. – 1024 с.

2. Эллис М., Страуструп Б. Справочное руководство по языку C++ с комментариями: Пер. с англ. - Москва: Мир, 1992.- 445с.

3. Ишкова Э.А. C++. Начала программирования – М.: ЗАО «Издательство БИНОМ», 2000. - 304 с.

4. Павловская Т.А. Программирование на языке высокого уровня С/С++. Учебник для студентов и преподавателей. Санкт-Петербург, 2002.

5. Стенли Б. Липпман. C++ для начинающих: Пер. с англ. 2т. - Москва: Унитех; Рязань: Гэлион. - 1992, 345с.

Приложение 1.

Пример оформления курсовой работы на тему «Массивы»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И.РАЗЗАКОВА

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРНЫХ СИСТЕМ

КУРСОВАЯ РАБОТА

ПО ДИСЦИПЛИНЕ: ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

НА ТЕМУ: МАССИВЫ

ВЫПОЛНИЛ: студент гр. ПОВТ
ПРОВЕРИЛ: _____

Бишкек 2009

СОДЕРЖАНИЕ КУРСОВОЙ РАБОБОТЫ:

1. Условие задачи.
2. Постановка задачи.
3. Графическое представление алгоритма решения в виде блок-схемы (блок-схема для каждой функции, в том числе и для main).
4. Словесный (пошаговый) алгоритм решения.
5. Программа на языке C++ .
6. Тестовые примеры.

---

1) Условие задачи:

Даны квадратная матрица и границы интервала.

Если значения максимального и минимального элементов матрицы не попадают на заданный отрезок [C,D], заменить значением D все элементы, которые больше D и заменить значением C все элементы, которые меньше C.

Найти сумму элементов новой матрицы.

2) Постановка задачи:

Считать из файла «massiv_start.txt» размер матрицы Matrix.size, диапазон [C,D] и квадратную матрицу Matrix.mas.

Найти минимальный и максимальный элементы матрицы min и max.

Если минимум и максимум матрицы не принадлежат отрезку [C,D], заменить на C все элементы, меньше C и заменить на D все элементы, больше D.

Вычислить сумму sum всех элементов матрицы.

Записать в файл «massiv_rezult.txt» итоговую матрицу и ее сумму.

Использовать динамические массивы, форматированный вывод данных, и если возможно, структуры.

Список используемых переменных:

Начальные данные:

Структура Matrix с параметрами size и mas.

Matrix.size – размер матрицы.

Matrix.mas – двумерный динамический массив (матрица).

C, D – границы отрезка [C, D].

Результирующие данные:

Matrix.mas – измененная матрица.

sum – сумма элементов измененной матрицы

Промежуточные данные:

min – минимальный элемент матрицы.

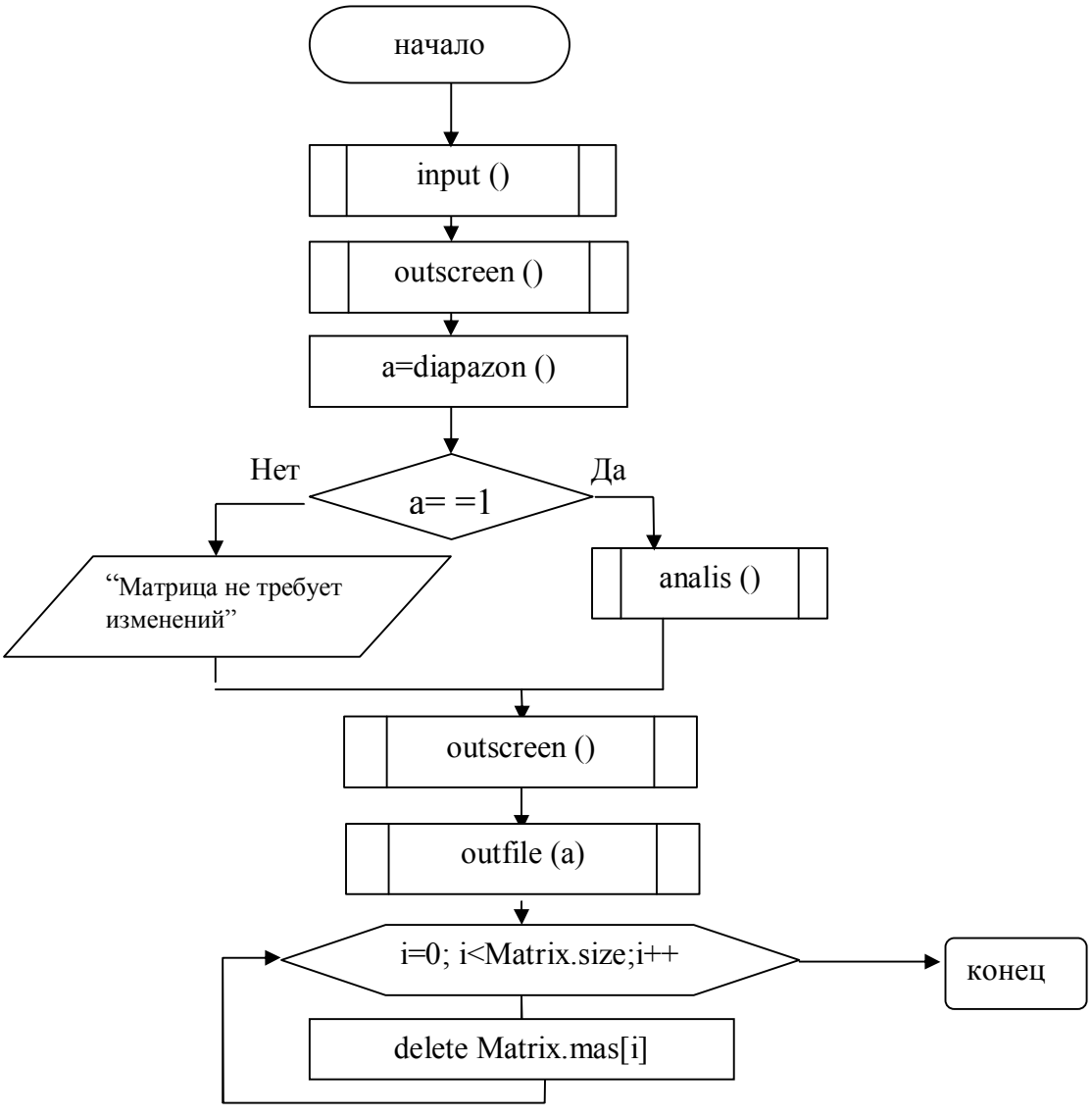
max – максимальный элемент матрицы.

i, j – счетчики цикла.

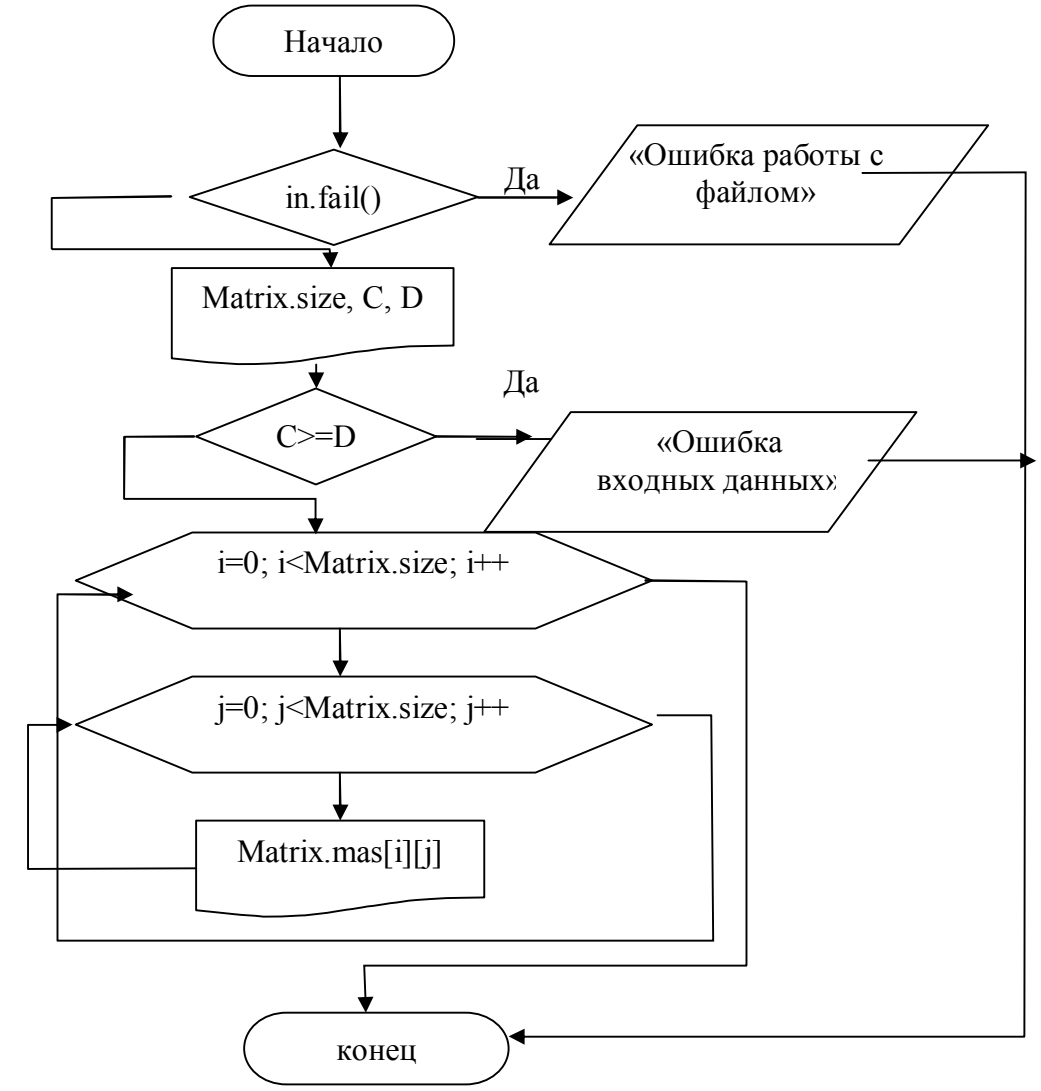
a –переменная для проверки необходимости изменения матрицы.

3) Блок-схемы

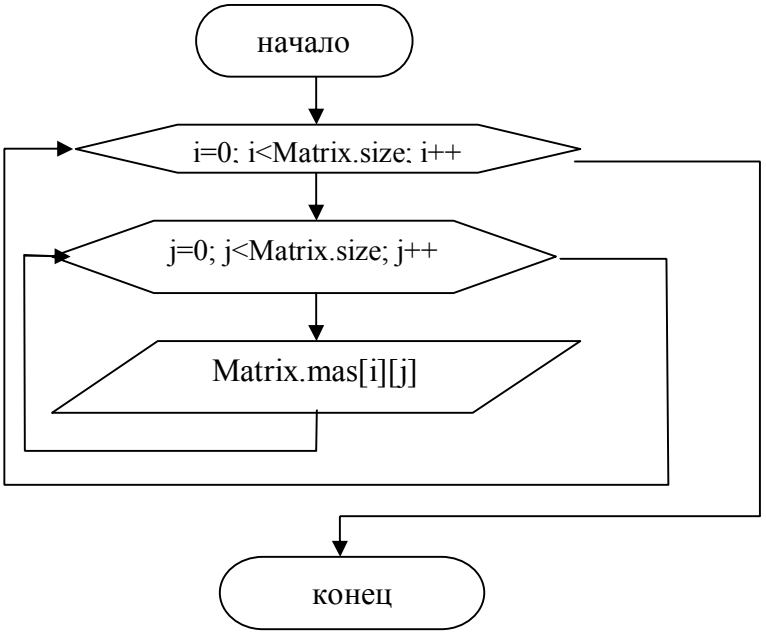
Функция main



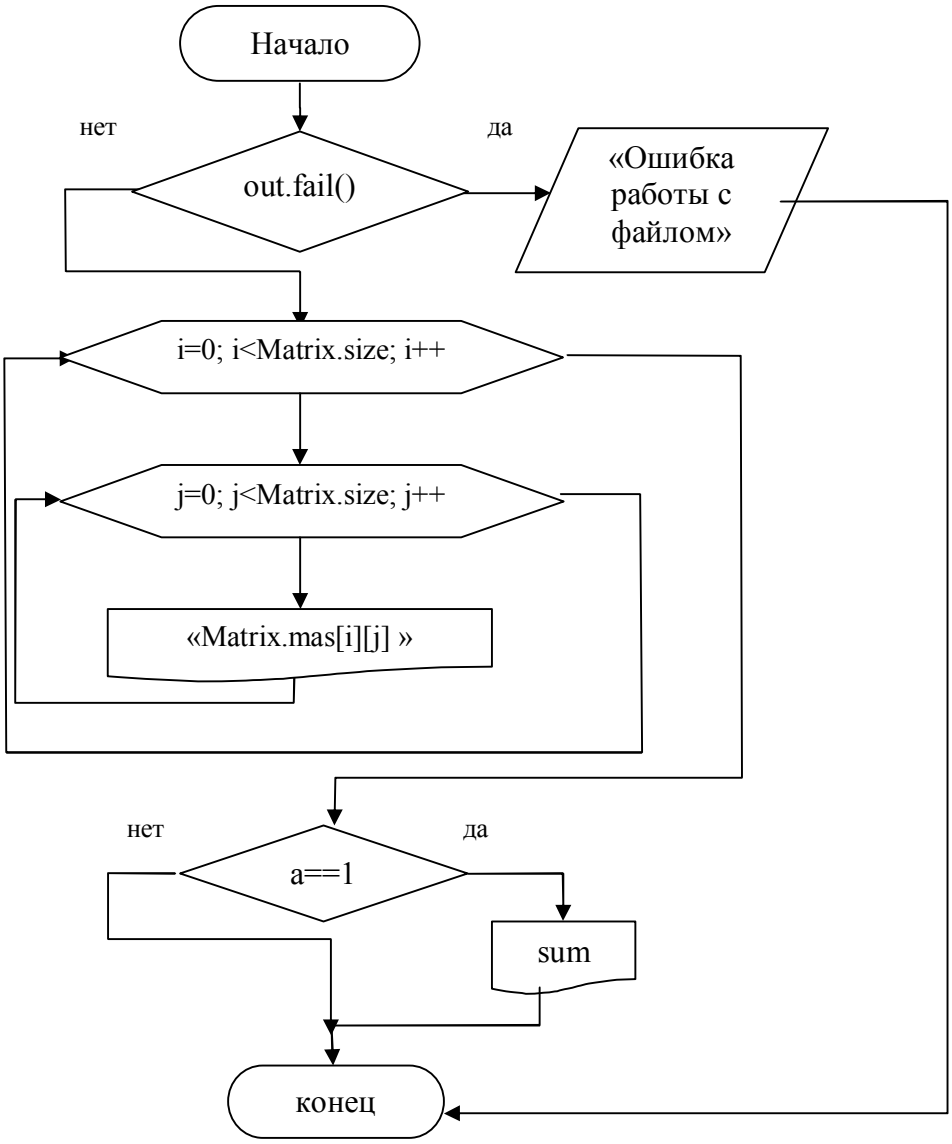
Функция input () - ввод матрицы и ее параметров из файла



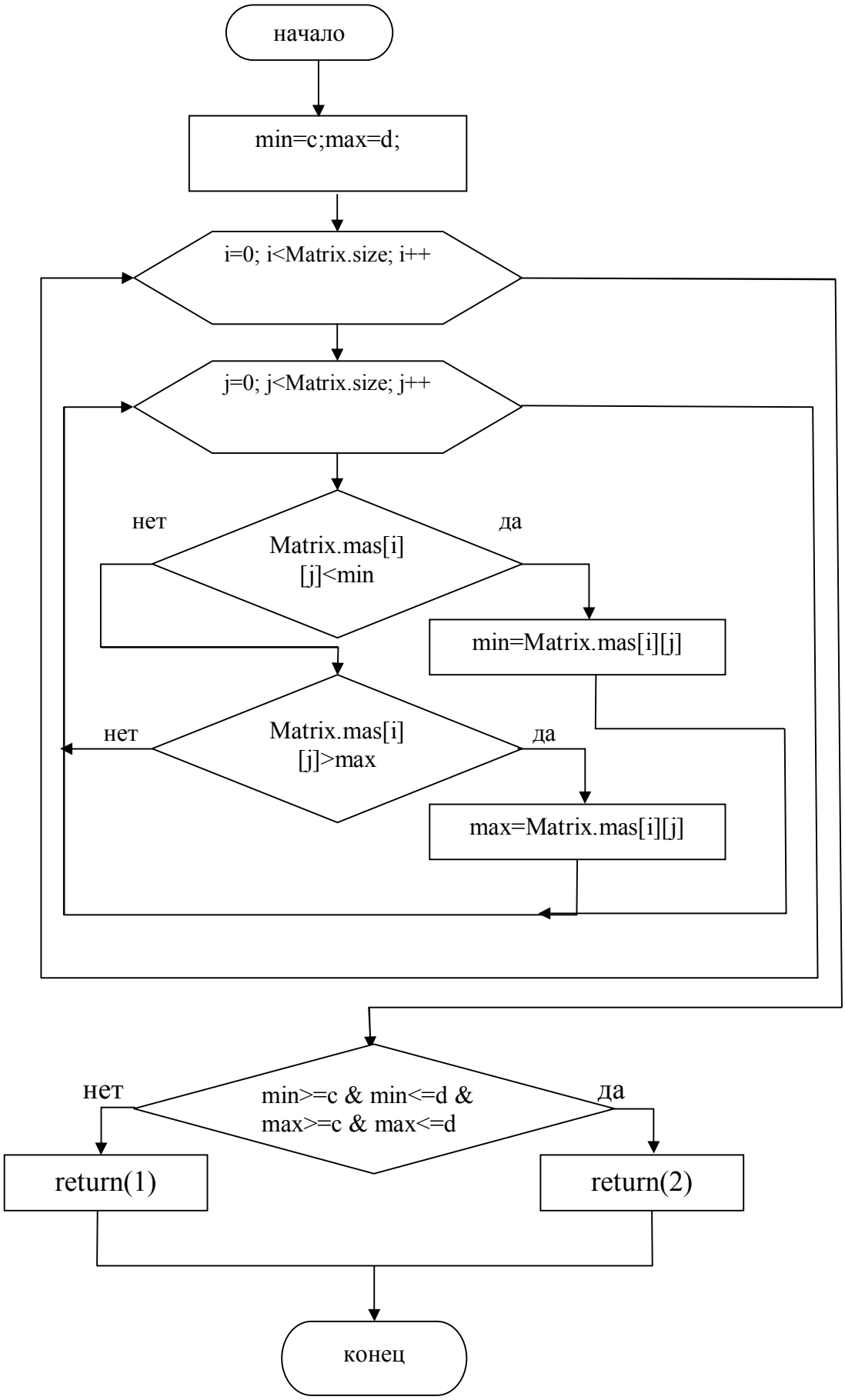
Функция outscreen () - вывод матрицы на экран



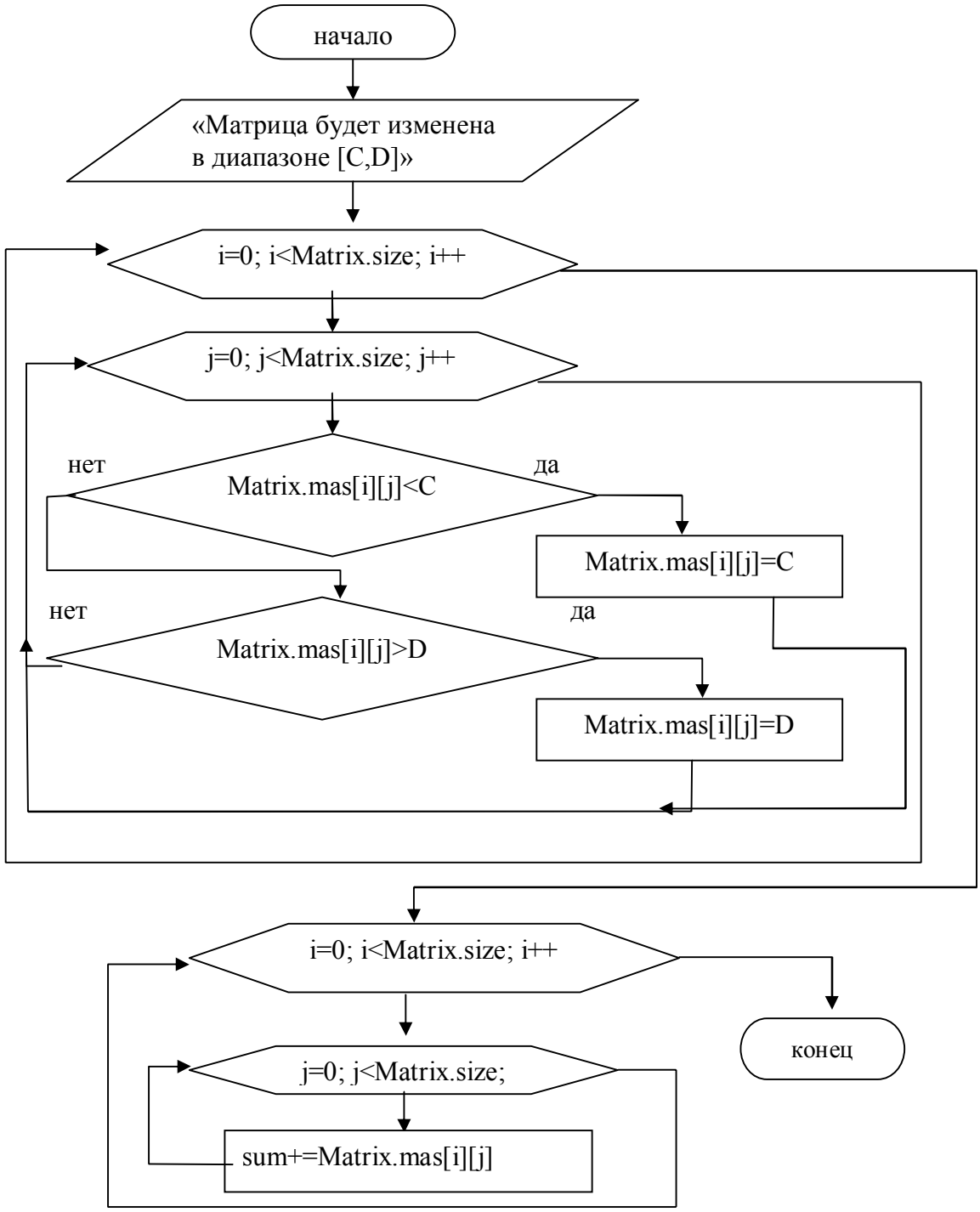
Функция outfile () - вывод результирующей матрицы в файл



Функция diapazon () - проверка необходимости анализа матрицы



Функция analis () - изменение матрицы в диапазоне [C,D]



4) Словесный (пошаговый) алгоритм решения:

Для того чтобы программа удовлетворяла требованиям к курсовой работе, она была реализована с использованием структур и динамических массивов. При реализации использовались функции из стандартных библиотек:

iostream.h - для вывода данных на экран;

fstream.h-для работы с файлами;

windows.h-для функции вывода русского текста.

Структура Matrix, содержащая размер матрицы и саму матрицу, переменные, обозначающие диапазон анализа, и переменная для суммы матрицы для упрощения работы программы и уменьшения количества передаваемой в функции информации, были объявлены как глобальные. Были разработаны 6 функций: main(), input(), diapazon(), outscreen(), outfile().

Все функции, кроме функции diapazon(), имеют тип void, т.е. не возвращают данные. Функция diapazon() имеет тип возвращаемого значения int и возвращает число.

Функция main() является главной и управляет последовательностью вызова остальных функций. Вначале вызывается функция input(). В ней объявляется входной поток in и открывается файл с данными для чтения. Затем функция считывает из файла размер матрицы и диапазон анализа. После этого создается динамический массив указанного размера и заполняется последовательным считыванием данных из файла. В конце файл закрывается и управление возвращается в главную функцию. Второй вызывается функция outscreen(). Эта функция выводит на экран считанную матрицу, проставляя табуляцию между всеми элементами и разделяя строки. После вывода управление возвращается в main().

Следующей вызывается функция diapazon(), значение которой присваивается промежуточной переменной ***a***. Эта функция находит минимальный и максимальный элементы матрицы, и проверяет, принадлежат ли они диапазону анализа. Если ответ положительный, то функция возвращает значение 2, если ответ отрицательный возвращает 1. После этого управление возвращается в функцию main().

Далее в main() проверяется значение переменной “a”. Если a=1, вызывается функция analis(). В функции analis() происходит проверка каждого элемента матрицы. Если элемент меньше нижнего предела диапазона “C”, он меняется на “C”. Если элемент больше верхнего предела диапазона “D”, он меняется на “D”. После изменения значений подсчитывается сумма “sum” всех элементов измененной матрицы и управление возвращается в функцию main(). Снова вызывается функция outscreen(), которая выводит на экран матрицу. Затем вызывается функция outfile(). В ней объявляется выходной поток out и открывается файл для записи. В него в форматированном виде записывается результирующая матрица и, если происходил анализ матрицы, печатается сумма “sum” элементов матрицы.

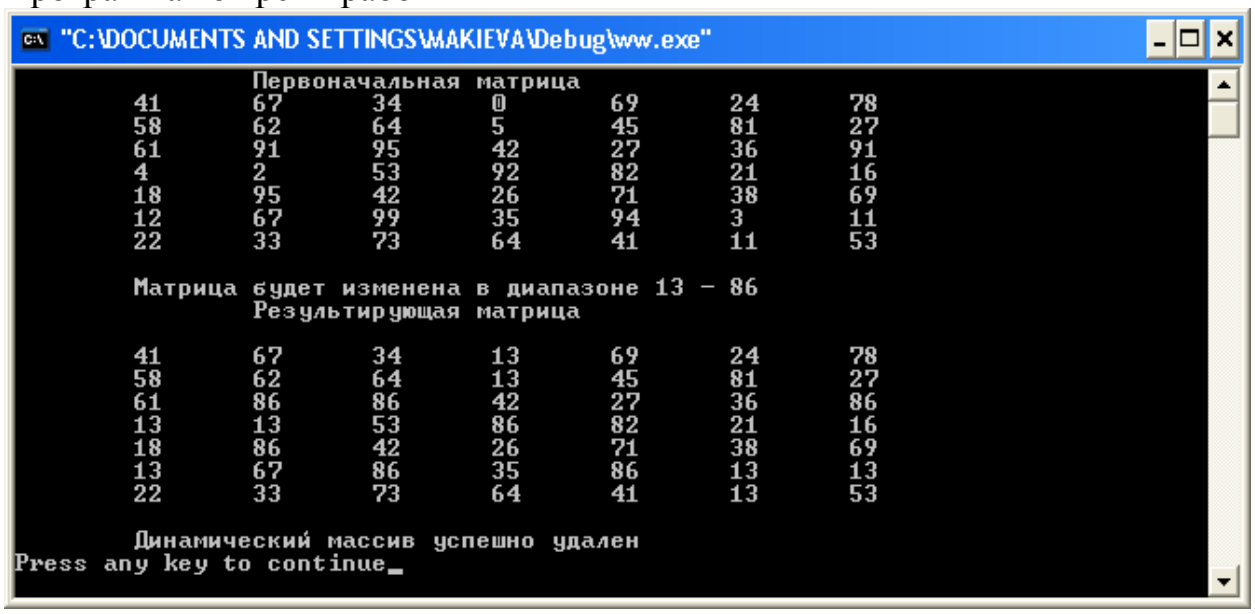
5) Программа на языке C++:

```
#include <iostream.h>
#include <fstream.h>
#include <windows.h>

char rustext[256];
char *rus(const char *text){CharToOem(text,rustext);
return rustext;};

struct mass{ int size;  double **mas; };
mass Matrix; int c,d;  double sum=0;

void input();
void outscreen();
int diapazon();
void analis();
void outfile(int a);

void main()
{ int a;
input();
cout <<rus("\n\n\t\tПервоначальная матрица\n\n\n");
outscreen();
a = diapazon();
if (a= =1) analis();
else cout <<rus("\t\tМатрица не требует изменений\n\n\n");
cout << rus("\t\tРезультирующая матрица\n\n\n");
outscreen();
outfile(a);
for (int i=0;i<Matrix.size;i++) delete Matrix.mas[i];
delete []Matrix.mas;
cout <<rus("\n\tДинамический массив успешно удален\n\n\n");} }

void input()
{  ifstream in;
        in.open("massiv_start.txt",ios::nocreate);
        if (in.fail()){cout << rus("Ошибка работы с файлом\n");  exit(1);}
        in >> Matrix.size >> c >> d;
        if (c>=d){cout <<rus("Ошибка входных данных 'C > D'"); exit (1);}
        Matrix.mas = new double *[Matrix.size];
        for (int i=0;i<Matrix.size;i++) Matrix.mas[i]=new double[Matrix.size];
        for (i=0;i<Matrix.size;i++){
                for (int j=0;j<Matrix.size;j++)   in >> Matrix.mas[i][j];};
        in.close();
}
```

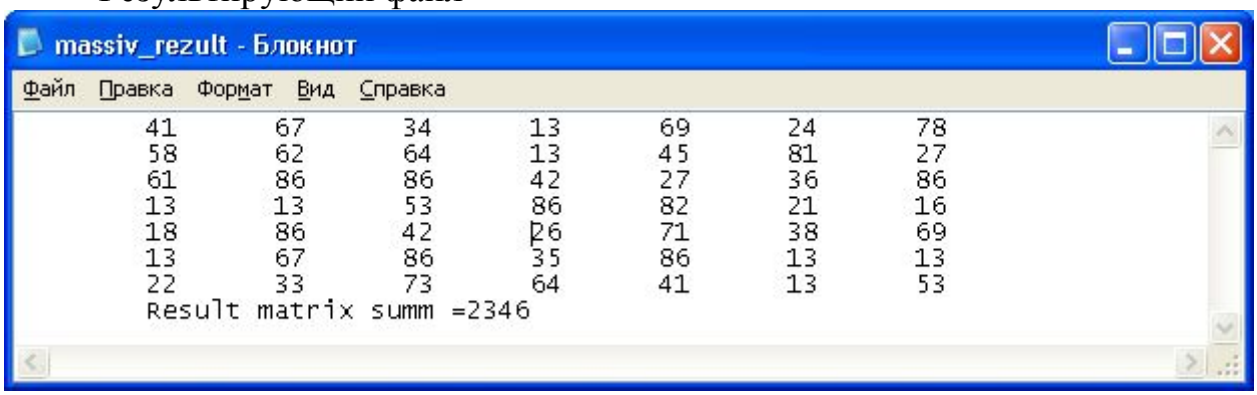
```
void outscreen()
{
	for (int i=0;i<Matrix.size;i++){cout<<"\t";
		for (int j=0;j<Matrix.size;j++)cout<<Matrix.mas[i][j]<<"\t";
		cout<<endl<<endl;}
}
int diapazon()
{
	double min=c, max=d;
	for (int i=0;i<Matrix.size;i++){
		for (int j=0;j<Matrix.size;j++){
			if (Matrix.mas[i][j]<min)min=Matrix.mas[i][j];
			else if (Matrix.mas[i][j]>max)max=Matrix.mas[i][j]; }}
	if (min>=c && min<=d && max>=c && max<=d) return(2);
	else return(1);
}
void analis()
{
cout << rus("\tМатрица будет изменена в диапазоне ")<<c<<" - "<<d<<"\n\n";
	for (int i=0;i<Matrix.size;i++){
		for (int j=0;j<Matrix.size;j++){
			if (Matrix.mas[i][j]<c)Matrix.mas[i][j]=c;
			else
				if (Matrix.mas[i][j]>d)Matrix.mas[i][j]=d;}  }
	for (i=0;i<Matrix.size;i++)
		for (int j=0;j<Matrix.size;j++) sum+=Matrix.mas[i][j];
}

void outfile(int a)
{
ofstream out;
out.open("massiv_rezult.txt");
for (int i=0;i<Matrix.size;i++){
for (int j=0;j<Matrix.size;j++) out << Matrix.mas[i][j]<<"\t"; out << endl;   }
if (a==1){    out <<"\tResult matrix summ ="<<sum;
out.close();}
 }
```

6) Тестовый пример: Первоначальный файл

Программа во время работы



Результирующий файл



Список литературы

1. Дж. Либерти Освой самостоятельно C++ за 21 день.
2. Дейтел Харви, Дейтел Пол. Как программировать на C++.