Министерство науки и высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра прикладной информатики в экономике и управлении

Н.Ф. Панова

ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ

Методические указания

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательной программе высшего образования по направлению подготовки 38.03.05 Бизнес-информатика

Оренбург
2018

УДК 004.42(076.5)
ББК 32.973я7
П 16

Рецензент - кандидат технических наук А.Н. Колобов

Панова, Н.Ф.
П 16 Программирование и основы алгоритмизации: методические указания / Н.Ф. Панова; Оренбургский гос. ун-т.- Оренбург : ОГУ, 2018. - 34 с.

Методические указания предназначены для выполнения курсовой работы по дисциплине «Программирование и основы алгоритмизации». Методические указания содержат теоретический материал по основам алгоритмизации и программирования, описание этапов выполнения курсовой работы, а также пример пояснительной записки к курсовой работе.

Предназначены для обучающихся по образовательной программе высшего образования по направлению подготовки 38.03.05 Бизнес-информатика очной и заочной форм обучения.

УДК 004.42(076.5)
ББК 32.973я7

Содержание

Введение

Дисциплина «Программирование и основы алгоритмизации» относится к базовой части блока 1 «Дисциплины (модули)», предусмотренных Государственным стандартом подготовки бакалавров по направлению 38.03.05 Бизнес-информатика.

Целью освоения дисциплины является формирование у будущих специалистов практических навыков по основам алгоритмизации вычислительных процессов и программированию решения экономических, вычислительных и других задач, развитие умения работы с персональным компьютером на высоком пользовательском уровне, обучение работе с научно-технической литературой и технической документацией по программному обеспечению ПЭВМ.

Задачами курса являются реализация требований, установленных в квалификационной характеристике, в подготовке специалистов в области использования вычислительной техники и ее программного обеспечения в системах машинной обработки экономической информации, проектирования и разработки этих систем.

Основными целями курсовой работы являются:

- систематизация и закрепление полученных теоретических знаний, полученных студентами при изучении дисциплины «Программирование и основы алгоритмизации»;

- углубление практических навыков;

- формирование умений применять теоретические знания при решении поставленных практических вопросов;

- раскрытие содержательной характеристики выбранной темы;

- приобретение и закрепление навыков самостоятельной работы;

- проверка умения формулировать основные выводы по результатам анализа конкретной темы.

Методические указания предназначены для выполнения курсовой работы по дисциплине «Программирование и основы алгоритмизации» для студентов первых курсов направления 38.03.05 Бизнес-информатика и содержат теоретический материал по теории и практике программирования на языке С.

1 Теоретические сведения

Существуют три вида вычислительных процессов:

- линейный;
- ветвящийся;
- циклический.

Линейным называется вычислительный процесс, этапы которого выполняются последовательно, в порядке их записи.

На блок - схеме для отображения линейного вычислительного процесса используется алгоритмическая структура «следование», изображенная на рисунке 1.

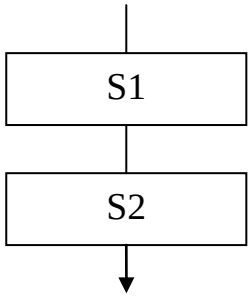


Рисунок 1 - Алгоритмическая структура «следование»

Вычислительный процесс называется ветвящимся, если в зависимости от исходных данных, желания пользователя или некоторого условия вычисления реализуются по тому или иному направлению. Эти направления называются ветвями вычислений.

В каждом конкретном случае выбирается только одна из ветвей, остальные игнорируются. Для отображения ветвящегося вычислительного процесса используется алгоритмическая структура «развилка», которая бывает полной и неполной.

Неполная развилка показана на рисунке 2.

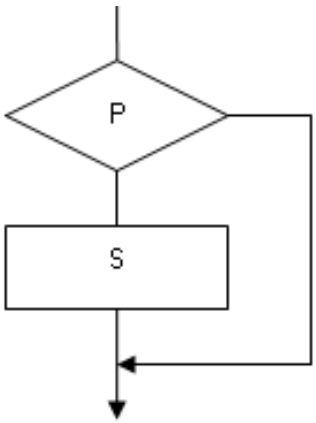


Рисунок 2 - Алгоритмическая структура «неполная развилка»

Здесь P – условие, S – произвольные алгоритмические структуры.

Полная развилка изображена на рисунке 3.

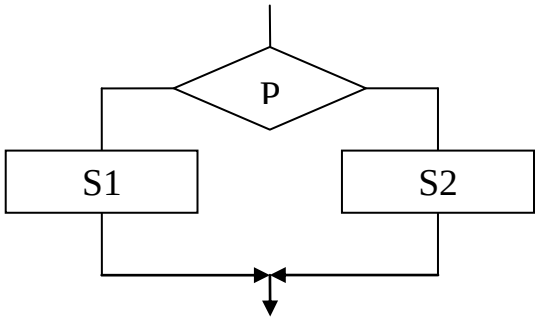


Рисунок 3 - Алгоритмическая структура «полная развилка»

Развилка называется вложенной, если блоки S1 или S2 или оба в свою очередь являются развилками. Пример вложенной развилки приведен на рисунке 4.

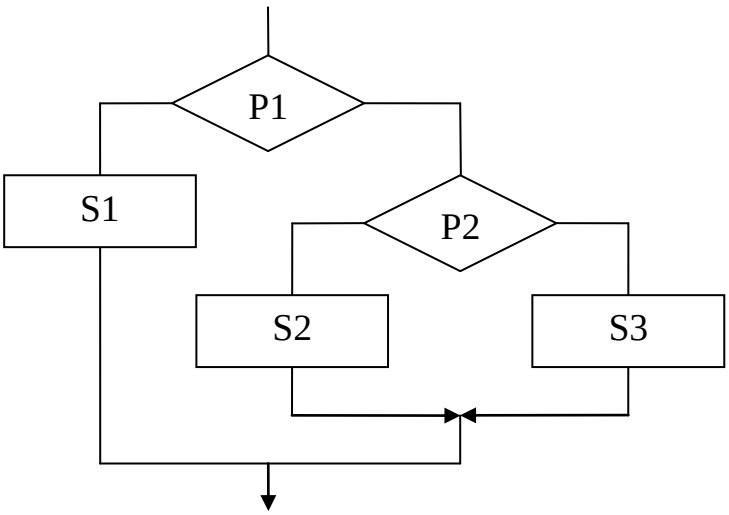


Рисунок 4 - Пример вложенной развилки

Для проверки большого количества условий используется алгоритмическая структура «переключатель», показанная на рисунке 5.

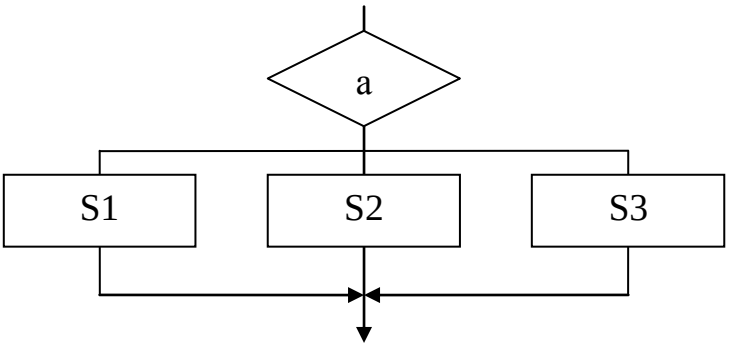
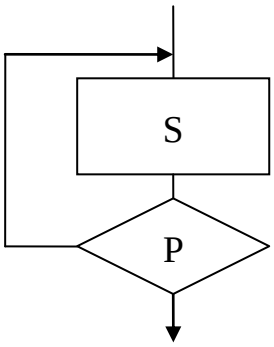


Рисунок 5 - Алгоритмическая структура «переключатель»

От значения переключателя «a» зависит выбор ветви вычислений.

Вычислительный процесс называется циклическим, если он содержит внутри себя повторяющиеся этапы, которые называются циклами.

Алгоритмические структуры для отображения циклов показаны на рисунках 6 и 7.



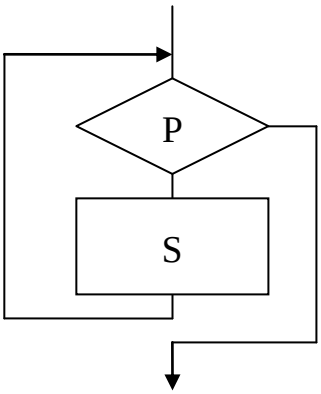


Рисунок 6 – Цикл с постусловием Рисунок 7 – Цикл с предусловием

Здесь P – условие выхода или продолжения цикла, S – тело цикла. Тело цикла может быть представлено последовательностью алгоритмических структур.

Цикл с предусловием может не выполниться ни разу, если условие P изначально означает выход из цикла. Цикл с постусловием должен выполняться хотя бы один раз.

В зависимости от числа повторений циклы делятся на циклы с известным числом повторений и циклы с неизвестным числом повторений (итерационные).

В циклах с известным числом повторений существуют следующие общие этапы:

1. Подготовительный. На этом этапе осуществляются некоторые начальные присваивания, т.е. происходит как бы подготовка к циклу, например, обнуляется переменная, в которой будет накапливаться сумма;

2. Инициализация параметра цикла;

3. Тело цикла;

4. Модификация параметра цикла;

5. Проверка условия выхода из цикла.

В качестве примера рассмотрим алгоритм табулирования функции $y = x^2 - 2 \cdot x + 1$, *где* $x \in [-2;2]$ *с шагом* 0,5.

В данном алгоритме параметром цикла является аргумент x. Начальное значение параметра согласно условию равно -2. Когда значение параметра превысит конечное значение, равное 2, произойдет выход из цикла. На каждом шаге значение параметра возрастает на 0.5. Так модифицируется параметр. В теле цикла вычисляется значение функции в точке x и на экран выводится пара значений – аргумент x и функция y.

На блок-схеме, изображенной на рисунке 8, использована базовая алгоритмическая структура «цикл с предусловием».

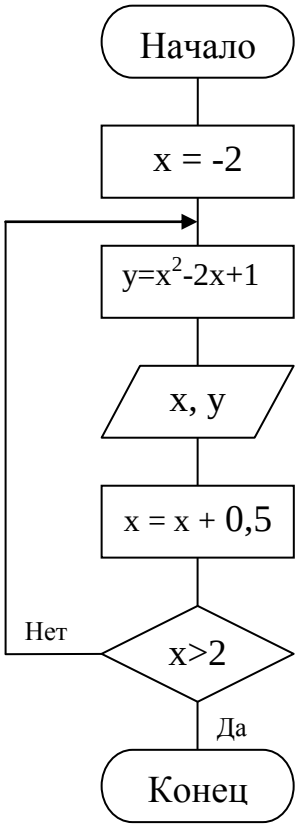


Рисунок 8 – Блок-схема алгоритма решения задачи

Циклы с неизвестным числом повторений или итерационные циклы параметра не имеют. Выход из таких циклов осуществляется при достижении заданной точности вычислений Eps (0<Eps<1).

В качестве примера рассмотрим вычисление $y = \sqrt{x}$ с использованием рекуррентного соотношения $y_n = \frac{1}{2}\left(y_{n-1} + \frac{x}{y_{n-1}} \right)$. Вычисления прекратить, когда выполнится условие $|y_n - y_{n-1}| < Eps$. Начальное приближение $y_0 = h$.

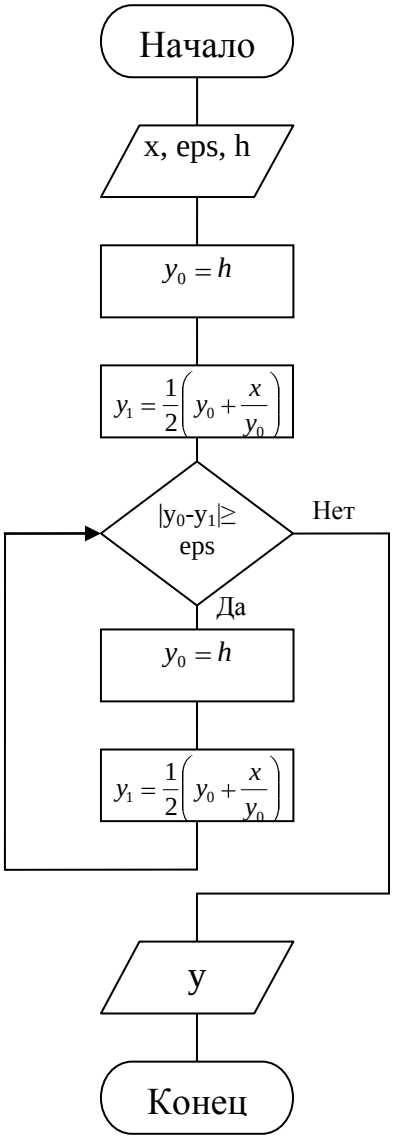


Рисунок 9 – Блок-схема алгоритма вычисления $y = \sqrt{x}$

2 Структура курсового проекта

Примерная структура курсового проекта:

- титульный лист;
- содержание;
- введение;
- постановка задачи;
- описание метода решения задачи;
- описание алгоритма решения задачи;

- текст программы;

- описание программы;

- анализ результатов прогона программы;

- заключение;

- список использованных источников.

Во введении обосновывается актуальность выбранной темы, ее важность. Формируются цели и задачи курсовой работы.

Содержание отражает основные разделы пояснительной записки к курсовой работе.

В разделе «Постановка задачи» определяются требования к программному продукту.

Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Для числовых данных может задаваться точность, для текстовых - возможно, размер текста, способ кодировки и т. п. Затем определяют перечень результатов, их характеристики и способы представления (в виде таблиц, диаграмм, графиков и т. п.). Кроме того, уточняют среду функционирования программного продукта: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

Рекомендуется в раздел «Постановка задачи» включить следующие подразделы:

- область применения. Здесь приводится назначение программы, формируются требования к входным данным – форма задания исходных данных (функций для вычислений, диапазон изменения параметров, погрешности вычислений, ограничения, например, отсутствие разрывов, наличие только одного экстремума и т.п.);

- требования к интерфейсу пользователя. В этой части следует определить, какие данные, в какой последовательности должны быть введены

для организации диалога с пользователем. Перечисляются виды и формы представления результатов работы программы;

- отдельно перечисляются возможные сообщения и реакция программы на ошибки ввода и вычислений. Приводятся тексты сообщений при возникновении ошибок.

Как правило, существует несколько методов решения задачи. При обосновании выбора метода необходимо учитывать различные факторы и условия, в том числе точность вычислений, время решения задачи на ЭВМ, требуемый объем памяти. В большинстве случаев метод решения задачи невозможно выбрать без предварительной математической формализации задачи, т.е. математического описания задачи.

Математическое описание задачи подразумевает описание исследуемого процесса или явления с помощью систем уравнений или неравенств. Математическая модель явления формулируется с определенной точностью, допущениями и ограничениями, поскольку реальные процессы и явления очень сложны и зависят от большого числа разнообразных факторов. При выполнении этого этапа могут быть использованы различные разделы математики, экономики и других дисциплин.

Математическая модель должна удовлетворять требованиям реалистичности и реализуемости. Другими словами, модель должна корректно отражать самые важные черты исследуемого объекта или процесса. При этом для разработанной модели должны быть известны методы решения или же была возможность разработать эвристический алгоритм.

Многие задачи могут быть описаны в терминах теории графов, в рамках которой разработано множество методов решения самых разнообразных задач. Для решения задач, в которых трудно или невозможно получить точное решение, широко используются численные методы, позволяющие получать решение приближенное.

Разработка алгоритма решения задачи осуществляется методом

пошаговой детализации. На первом этапе алгоритм представляется в виде нескольких обобщенных этапов, которые постепенно детализируются до получения готового к исполнению алгоритма. Алгоритм описывается с помощью блок-схем в соответствии с требованиями ГОСТ ЕСПД.

При описании программы указываются общие сведения о программе - обозначение и наименование программы, программное обеспечение, необходимое для функционирования программы, языки программирования, на которых написана программа.

Далее следует описать логическую структуру программы, т.е. ее составные части (функции) и связи между ними. Описание логической структуры программы выполняют с учетом текста программы на исходном языке.

Можно пояснить назначение некоторых переменных или подробности выполнения отдельных фрагментов. Кроме того, необходимо описать, как реализован в программе пользовательский интерфейс, какой пункт меню следует выбрать или какое значение ввести с клавиатуры для выполнения того или иного действия.

Разработанную программу необходимо протестировать на различных наборах исходных данных. Тестовые наборы должны быть составлены таким образом и в таком количестве, чтобы можно было проверить как можно больше маршрутов выполнения программы. Например, предусмотреть случаи, когда задача имеет только одно решение, не имеет решения вовсе и имеет множество решений. Результат каждого теста необходимо представить в пояснительной записке и прокомментировать или сделать вывод.

В заключении студент формулирует основные выводы по теме и рекомендации относительно возможностей практического применения материалов работы. Следует также указать на имеющиеся проблемы и возможные перспективы их решения.

В списке использованных источников перечисляется литература, использованная при выполнении курсовой работы, в том числе и ГОСТы.

В приложениях можно поместить скрин-шоты экранов, экранные формы программы, блок-схемы алгоритмов и т.д.

3 Пример разработки курсовой работы

В качестве примера рассмотрим курсовую работу на тему «Анализ больших текстов»

3.1 Введение

Анализ больших текстов в настоящее время приобрел большое значение в связи с ростом конкуренции на рынке информационных ресурсов и услуг. Только конкурентоспособные, качественные сайты и порталы могут привлечь посетителей и тем самым повысить эффективность бизнеса. Показателем качественности СЕО-текста, который составляется и размещается на страницах сайта с целью его продвижения, является семантический анализ. Написание текста для оптимизации требует наличия в нем ряда показателей, которые не должны превышать соответствующие нормы.

Показатели семантического анализа можно разделить на следующие категории:

- тошнота ключевых слов - частота употребления в тексте слов, составляющих семантическое ядро. Запросы, по которым продвигается страница, по частоте приведения в тексте должны преобладать над всеми остальными словами, но в целом их соотношение к общей массе слов не должно превышать 6%;

- стоп-слова - количество в тексте слов, не несущих смысловой нагрузки (предлоги, союзы, местоимения, наиболее часто употребительные в интернете существительные, глаголы и др.). Стоп-слова относятся к так называемой воде;

- вода - совокупность в тексте незначимых слов и выражений. Соотношение воды и общего количества слов в тексте не должно превышать

30-40%, в противном случае текст теряет свою значимость, становится малосодержательным и неинтересным для читателя.

Целью данной курсовой работы является создание программы, реализующей фрагмент семантического анализа большого текста, а именно: расчет статистических характеристик текста в наглядном виде.

3.2 Постановка задачи

Исходный текст должен храниться на диске. Текст произвольный и имеет любой размер. Программа считывает построчно информацию из файла, анализирует символы в каждой строке, рассчитывает частоты появления различных символов в тексте и длины слов. В качестве результата, программа должна выводить статистику распределения частоты употребления букв, распределения длин слов и знаков препинания.

3.3 Описание алгоритма решения задачи

Ниже представлено описание алгоритма в словесной форме:

1. Начало.

2. Ввод имени файла. Перейти к шагу 3.

3. Если файл не найден, вывести сообщение «Ошибка открытия файла» и завершить программу, иначе перейти к шагу 4.

4. Пока не достигнут конец файла, считывать содержимое файла в строку длиной 200 символов и анализировать символы в ней, а также подсчитывать количество слов в каждой строке. Перейти к шагу 5.

5. Считать команду. Перейти к шагу 6.

6. Если команда A вывести статистику латинских букв, если команда B вывести статистику русских букв, если команда C вывести статистику знаков препинания. Перейти к шагу 7. Если введен неверный символ, вывести на экран сообщение «Неизвестная операция», завершить программу.

7. Закрыть файл.

8. Конец.

Блок-схема алгоритма представлена в приложении А.

3.4 Описание программы

Для написания программы использовался язык Си и среда разработки Code::Blocks. Для работы с файлами в Си необходимо использовать специальные функции, определенные в заголовочном файле stdio.h. Открытый файл является последовательностью считываемых данных, поэтому при открытии файла с ним связывается поток ввода-вывода. Вся информация из файла записывается в поток ввода-вывода.

Поток связывается со стандартной структурой FILE, которая будет содержать информацию о файле.

Открытие файла осуществляется с помощью функции fopen(), тогда структура будет иметь следующий вид: которая возвращает указатель на структуру типа FILE, который можно использовать для последующих операций с файлом.

FILE *f=fopen(name,type);

где name – имя открываемого файла,

type – указатель на строку символов, определяющих доступ к файлу.

Для текущей задачи нужно выбрать указатель "r" – открытие файла для чтения.

Если обнаружена ошибка, то возвращается значение NULL, открыть файл невозможно.

Для чтения символов используется функция *fgets(УказательНаСтроку,КоличествоСимволовВстроке,поток)*;

Символы считываются до тех пор, пока не будет достигнут конец строки или пока не наступит конец потока EOF. Функция feof(поток) проверяет, достигнут ли конец файла, связанного с потоком.

Чтобы закрыть файл используется функция fclose(поток);.

Анализ строк производится с помощью цикла с параметром.

Для изменения общего фона и цвета текста консоли вызывают функцию *system("color<A><B>")*, где *<A>* - цвет фона и *<B>* - цвет текста. Например, для установки белого фона и черного текста вызывается *system("color F0").*

Для изменения цвета отдельных слов необходимо подключить заголовочный файл *<windows.h>* и получить дескриптор консоли с помощью следующей строки: *HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE)*. Затем перед строкой, выводящей текст, цвет которого нужно изменить, вызываем функцию *SetConsoleTextAttribute(hConsole, (WORD) ((<цвет фона за этим текстом><< 4) | <цвет текста >))*. Например, текст после следующей строки *SetConsoleTextAttribute(hConsole, (WORD) ((14<< 4) | 10));* будет зеленым на желтом фоне.

В таблицах 1 и 2 поясняются использованные функции и переменные:

Таблица 1 – Описание функций

Тип	Имя функции	Описание функции
int	main()	Главная функция
void	zap1(*sim)	Заполняет массив sim символами из кодировочной таблицы
void	kolvosimvolov(*s,*sim,*kolvosim)	Подсчитывает частоту использования каждого символа массива-алфавита
void	slova(*s,*slovar)	Подсчитывает количество букв в словах и считает частоту использования слов разной длины
void	printlat(*sim,*kolvosim,*dkolvosim)	Выводит статистику использования латинских символов
void	printrus(*sim,*kolvosim,*dkolvosim)	Выводит статистику использования русских символов
void	printsim(*sim,*kolvosim,*dkolvosim)	Выводит статистику использования знаков препинания
void	printslova(*slovar)	Выводит частоту употребления слов разной длины

Таблица 2 – Описание переменных

имя переменной	Тип	Описание переменной
name[200]	массив char	Хранит имя файла
s[200]	массив char	Хранит информацию из файла
sim[256]	массив char	Хранит алфавит
kolvosim[256]	массив int	Хранит частоту символов
dkolvosim[256]	массив int	Хранит значения частот в масштабе (используется для построения диаграммы)
slovar[20]	массив int	Хранит частоту использования слов разных длин
max	int	Максимальная частота
i,j,c	int	Счетчик

Блок-схемы алгоритмов, реализуемых функциями, представлены в приложении Б.

3.5 Текст программы

```
#include<conio.h>
#include<stdio.h>
#include<locale.h>
#include<string.h>
#include <windows.h>

int main()
{
   HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
setlocale(LC_ALL,"Rus");
system("color 9F");
   char name[200];
   char s[200];
   char sim[256];
   int i,kolvosim[256],dkolvosim[256],slovar[20];
   zap1(&sim);
   for (i=0;i<256;i++)
      {
kolvosim[i]=0;
dkolvosim[i]=0;
      }
   for (i=0;i<20;i++)
slovar[i]=0;
SetConsoleTextAttribute(hConsole, (WORD) ((9 << 4) | 14));
printf("Введите имя файла: ");
```

```
SetConsoleTextAttribute(hConsole, (WORD) ((9 << 4) | 15));
   gets(name);
   FILE *f = fopen(name,"r");
   if (f == NULL)
   {
SetConsoleTextAttribute(hConsole, (WORD) ((9 << 4) | 12));
printf("Ошибка открытия файла.\n");
exit(1);
   }
   while(!feof(f))
   {
fgets(s,200,f);
kolvosimvolov(&s,&sim,&kolvosim);
slova(&s,&slovar);
   }
SetConsoleTextAttribute(hConsole, (WORD) ((9 << 4) | 14));
printf("\nВывести статистику:\n\nA – латинских символов;\nB – русских символов;\nC – знаков препинания;\nD - длинслов.\n\n");
   char command;
SetConsoleTextAttribute(hConsole, (WORD) ((9 << 4) | 15));
   command = getchar();
   switch(command)
{ case 'A': case 'a':case 'A':case 'a': printlat(&sim,&kolvosim,&dkolvosim);break;
      case 'B': case 'b': case 'B': printrus(&sim,&kolvosim,&dkolvosim);break;
      case 'C': case 'c': case 'C': case 'c': printsim(&sim,&kolvosim,&dkolvosim);break;
      case 'D': case 'd': printslova(20,&slovar);break;
      default: {SetConsoleTextAttribute(hConsole, (WORD) ((9 << 4) | 12)); printf("Неизвестная операция.");};break;
   }
fclose(f);
return 0;
}

void zap1(char *sim)
{
   int i;
   for (i=0;i<256;i++)
      sim[i]=(char)i;
}

void kolvosimvolov(char *s, char *sim, int *kolvosim)
{
```

```
   int i,j;
   for (i=0;i<=strlen(s);i++)
      for (j=32;j<256;j++)
      if (s[i]==sim[j]) kolvosim[j]++;
}

void slova(char *s, int *slovar)
{
   int i,j,c=0;
   for (i=0;i<strlen(s);i++)
   if (s[i]!=' ') c++; else {
                             for (j=0;j<20;j++)
                             if (j+1==c) slovar[j]++;
                             c=0;
                           }

}
void printlat(char *sim, int *kolvosim, int *dkolvosim)
{
inti,j,max;
printf("\nЧастота использования латинских букв:\n");
max=kolvosim[65];
   for (i=65;i<123;i++)
      if(kolvosim[i]>max) max=kolvosim[i];
   for (i=65;i<123;i++)
dkolvosim[i]=(kolvosim[i]*70)/max;
   for (i=65;i<123;i++)
   {
      if (i==90) i=97;
printf("%c|",sim[i]);
      for(j=1;j<=dkolvosim[i];j++)
printf("=");
printf (" %d",kolvosim[i]);
printf("\n");
   }

}

void printrus(char *sim, int *kolvosim, int *dkolvosim)
{
inti,j,max;
printf("\nЧастота использования русских букв:\n");
max=kolvosim[192];
   for (i=192;i<256;i++)
```

```
        if(kolvosim[i]>max) max=kolvosim[i];
    for (i=192;i<256;i++)
dkolvosim[i]=(kolvosim[i]*70)/max;
    for (i=192;i<256;i++)
    {
printf("%c|",sim[i]);
        for(j=1;j<=dkolvosim[i];j++)
printf("=");
printf (" %d",kolvosim[i]);
printf("\n");
    }}

void printsim(char *sim, int *kolvosim, int *dkolvosim)
{
inti,j,max;
printf("\nЧастота использования знаков препинания:\n");
max=kolvosim[32];
    for (i=32;i<96;i++)
        if(kolvosim[i]>max) max=kolvosim[i];
    for (i=32;i<96;i++)
dkolvosim[i]=(kolvosim[i]*70)/max;
    for (i=32;i<96;i++)
    {
        if (i==48) i=58;
        if (i==65) i=95;
printf("%c|",sim[i]);
        for(j=1;j<=dkolvosim[i];j++)
printf("=");
printf (" %d",kolvosim[i]);
printf("\n");
    }
}

void printslova(int n,int *slovar)
{
int i;
printf("\nСтатистика длин слов:");
for (i=0;i<20;i++)
printf("\nСлов из %d букв - %d",i+1,slovar[i]); }
```

3.6 Анализ результатов прогона программы

На рисунках 10 и 11 представлены виды экранов с результатами выполнения программы, когда пользователь вводит символ «A» для вывода статистики использования в тексте латинских букв.

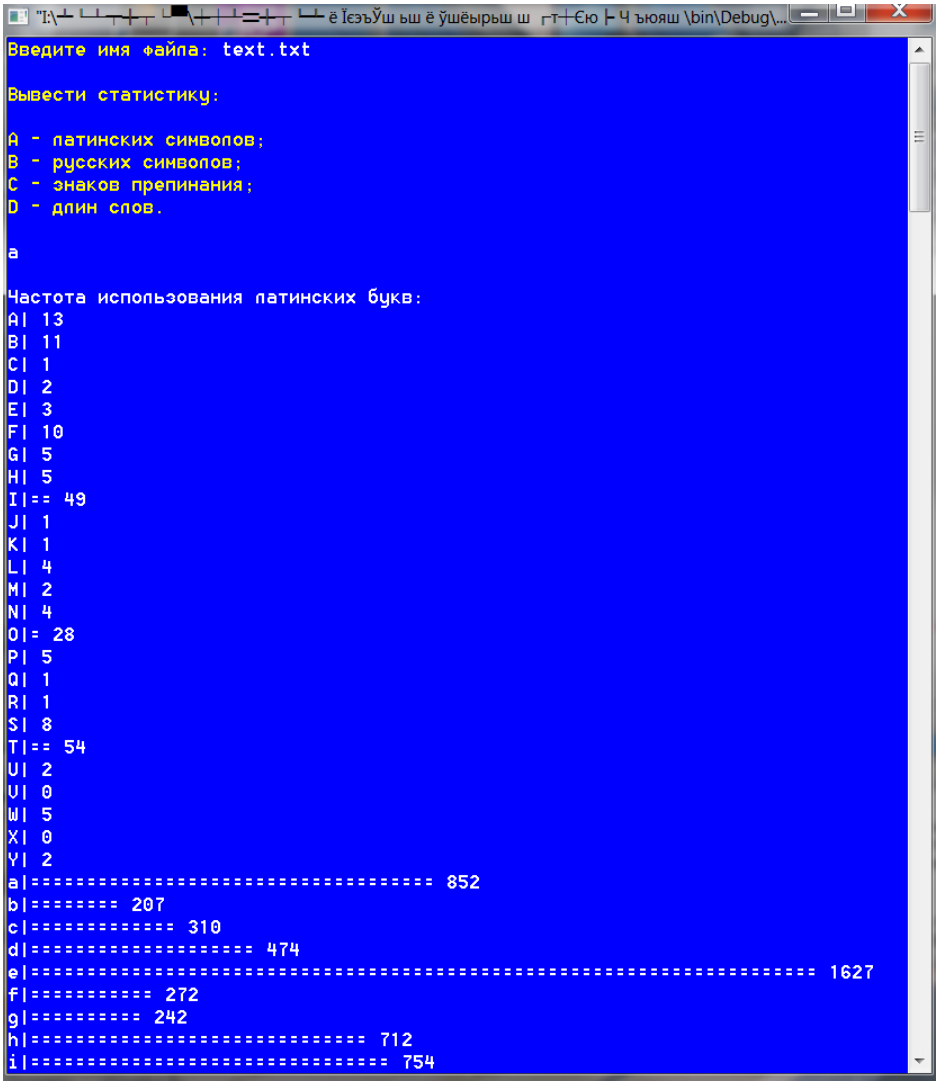


Рисунок 10 – Вид экрана после вычисления частот использования латинских букв в тексте

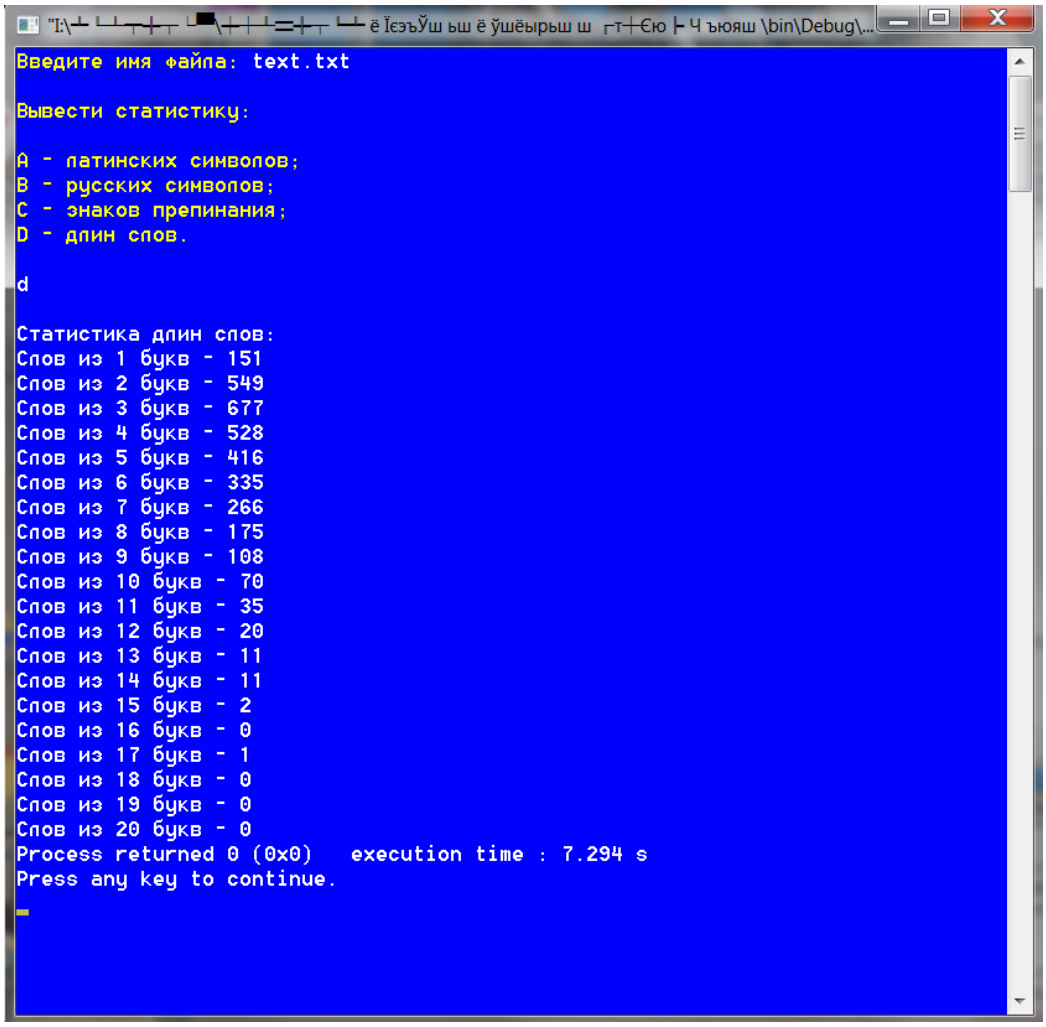


Рисунок 11 – Вид экрана с результатами анализа длин слов

Очевидно, что анализируемый текст не оптимизирован и содержит высокий процент «воды», поскольку в него входит 549 слов, состоящих из 2 букв, а это могут быть только предлоги или союзы.

Заключение

Результатом выполнения курсовой работы является программа, написанная на языке С, реализованная в среде разработки Code::Blocks. Это свободная кроссплатформенная среда разработки, имеющая открытую архитектуру. Программа функционирует под управлением операционной системы Windows и представляет собой консольное приложение.

Программа в дальнейшем может быть доработана путем реализации таких основных функций СЕО-анализа, как расчет воды и плотности кдючевых слов.

Список использованных источников

1 Подбельский, В. В. Программирование на языке Си [Текст] : учеб. пособие для вузов / В. В. Подбельская, С. С. Фомин.- 2-е изд., доп. - М. : Финансы и статистика, 2003. - 600 с. : ил. - ISBN 5-279-02180-6

2 Седжвик, Р. Фундаментальные алгоритмы на С [Текст] : пер. с англ. / Р. Седжвик . - 3-я ред. - М. ; СПб. ; Киев : Торгово-издат. Дом "DiaSoft", 2003. Ч. 5 : Алгоритмы на графах. - 480 с. : ил. - Предм. указ.: с. 1121-1127. - ISBN 5-93772-082-2. Режим доступа

3 Павловская, Т. А. C/C++. Программирование на языке высокого уровня [Текст] : для магистров и бакалавров: учебник для студентов высших учебных заведений, обучающихся по направлению подготовки дипломированных специалистов "Информатика и информационная техника" / Т. А. Павловская. - Санкт-Петербург : Питер, 2013. - 461 с. : ил. - (Учебник для вузов. Стандарт третьего поколения). - Библиогр.: с. 383. - Прил.: с. 384-449. - Алф. указ.: с. 450-460. - ISBN 978-5-496-00031-4.

4 Хусаинов, Б. С. Структуры и алгоритмы обработки данных. Примеры на языке Си [Комплект] : учеб. пособие / Б. С. Хусаинов. - М. : Финансы и статистика, 2004. - 464 с. : ил + 1 электрон. опт. диск (CD-ROM). - Библиогр.: с. 462-464. - ISBN 5-279-02775-8.

Приложение А
Блок-схемы алгоритмов

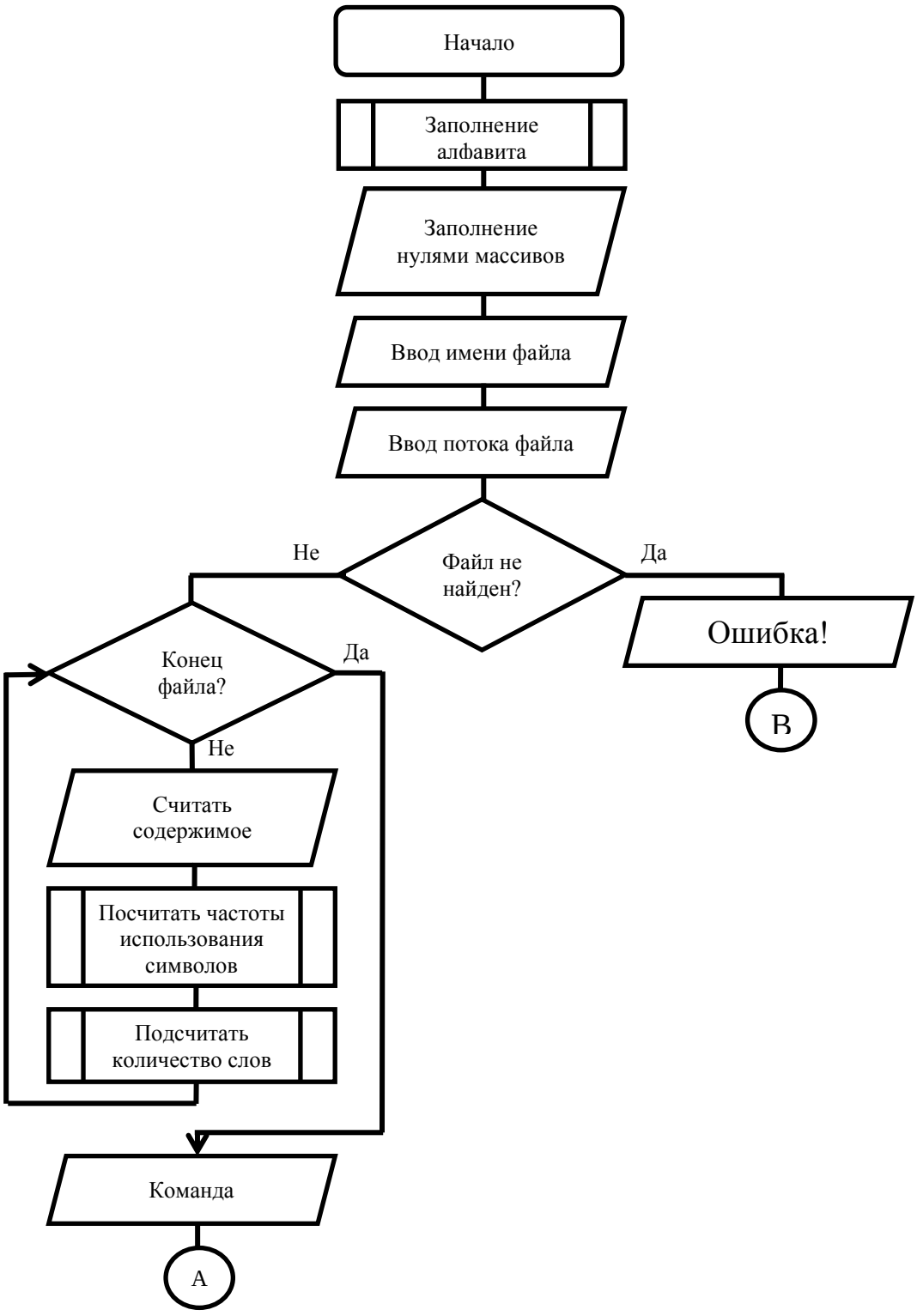


Рисунок А.1 - Обобщенная блок-схема алгоритма решения задачи

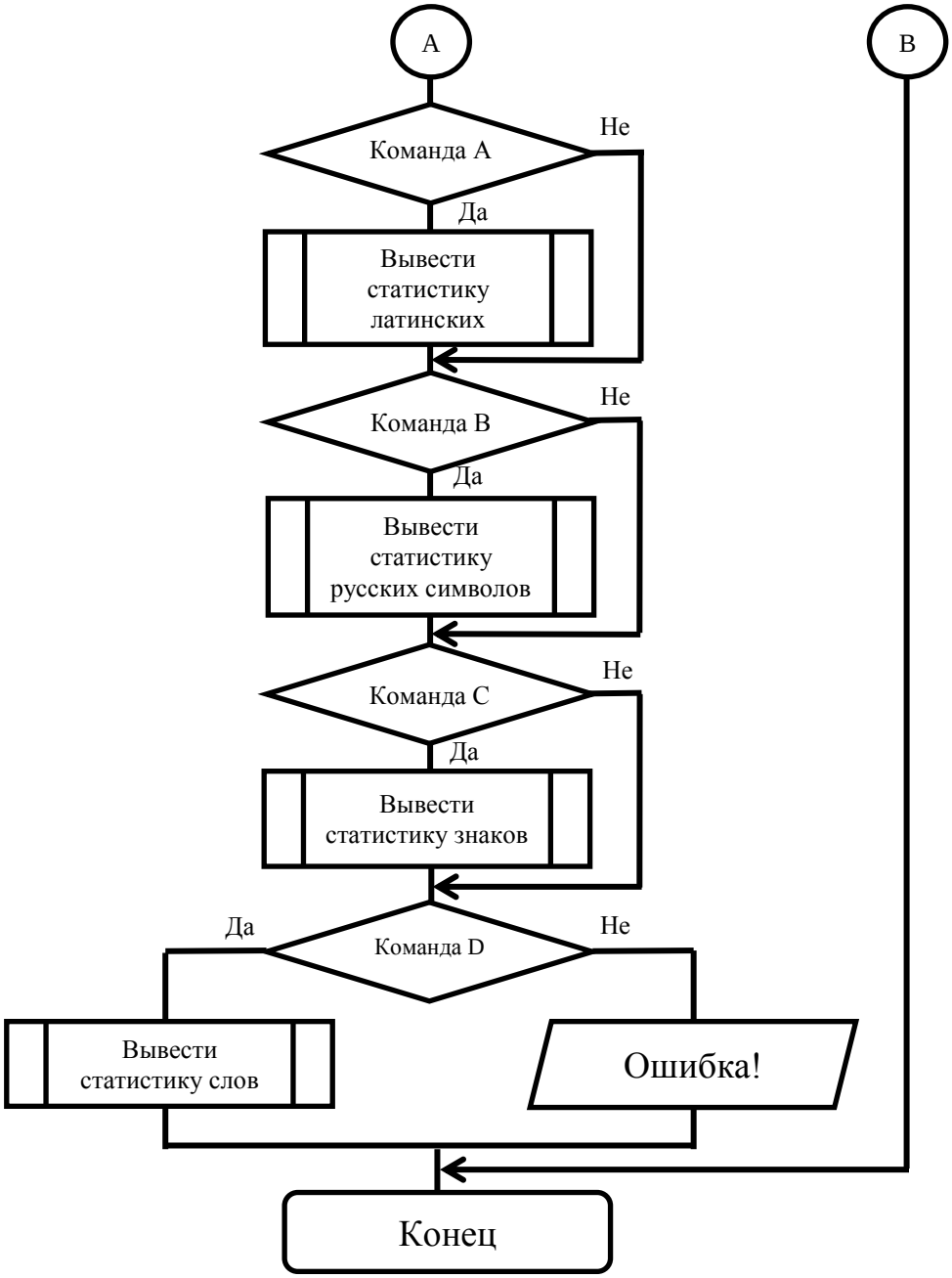


Рисунок А.2 - Продолжение обобщенной блок-схемы алгоритма решения задачи

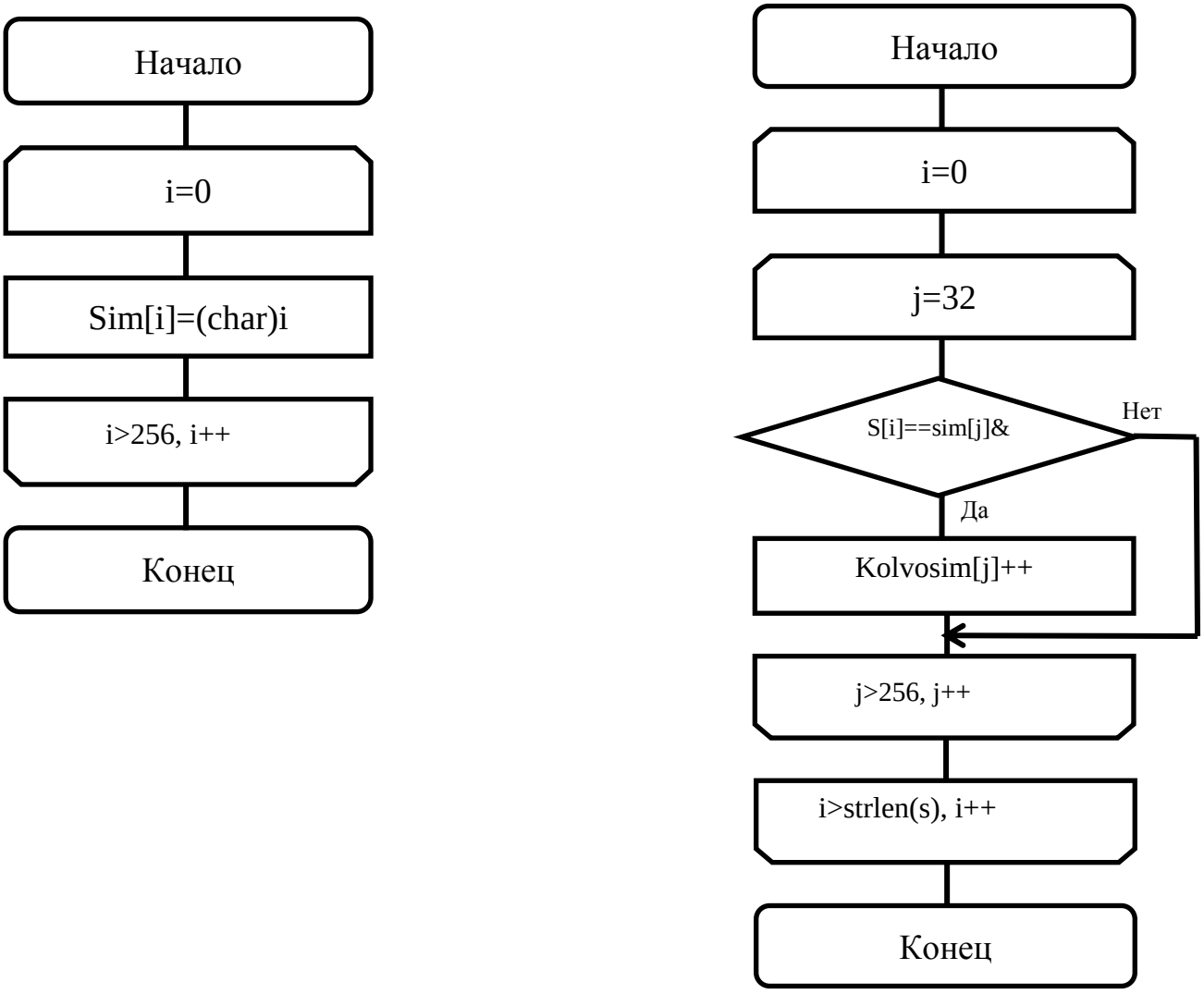


Рисунок А.3 – Блок-схемы алгоритмов заполнения вспомогательного массива и подсчета количества символов

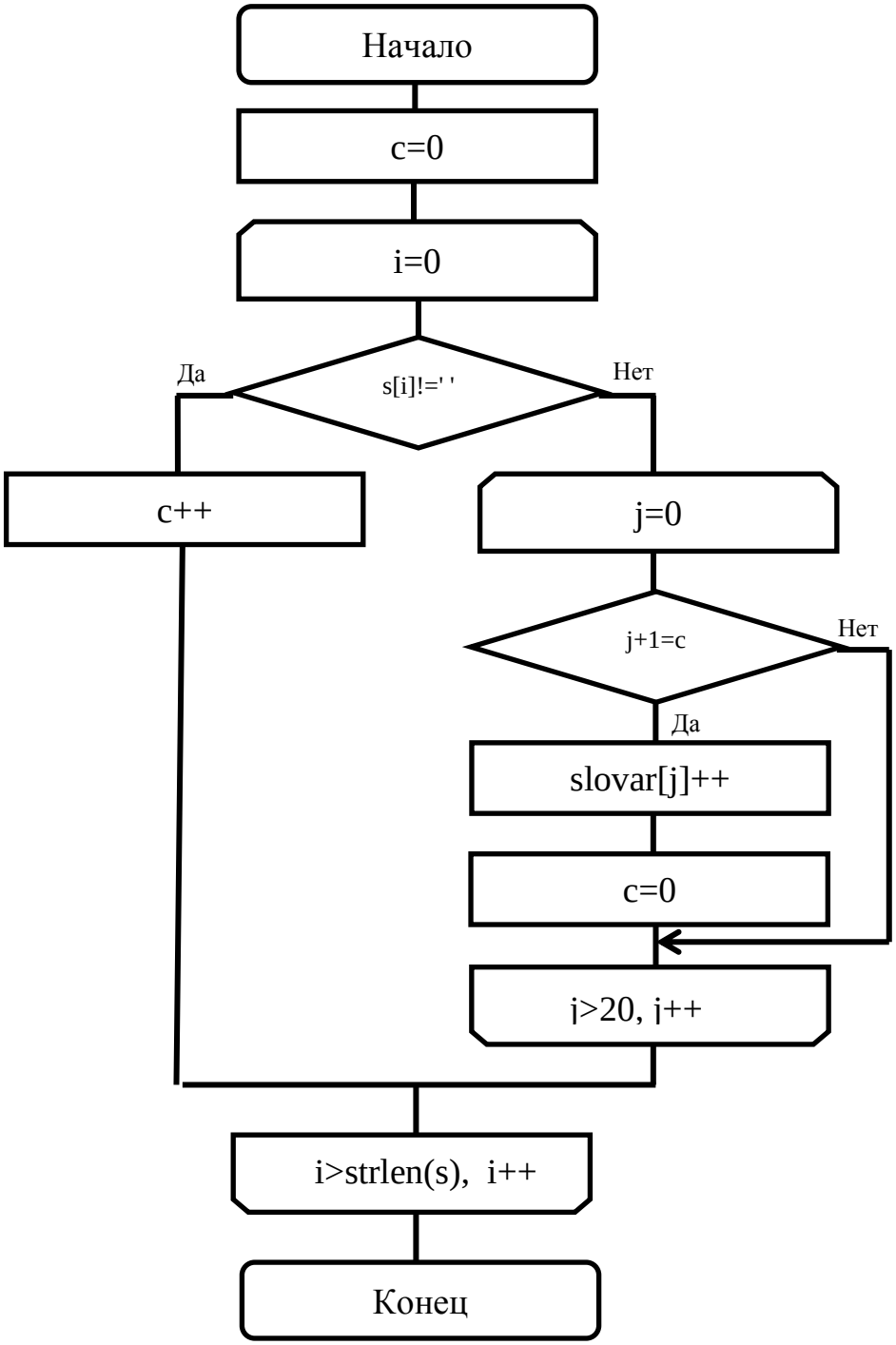


Рисунок А.4 - Блок-схема алгоритма подсчета количества букв в словах и частоты использования слов разной длины

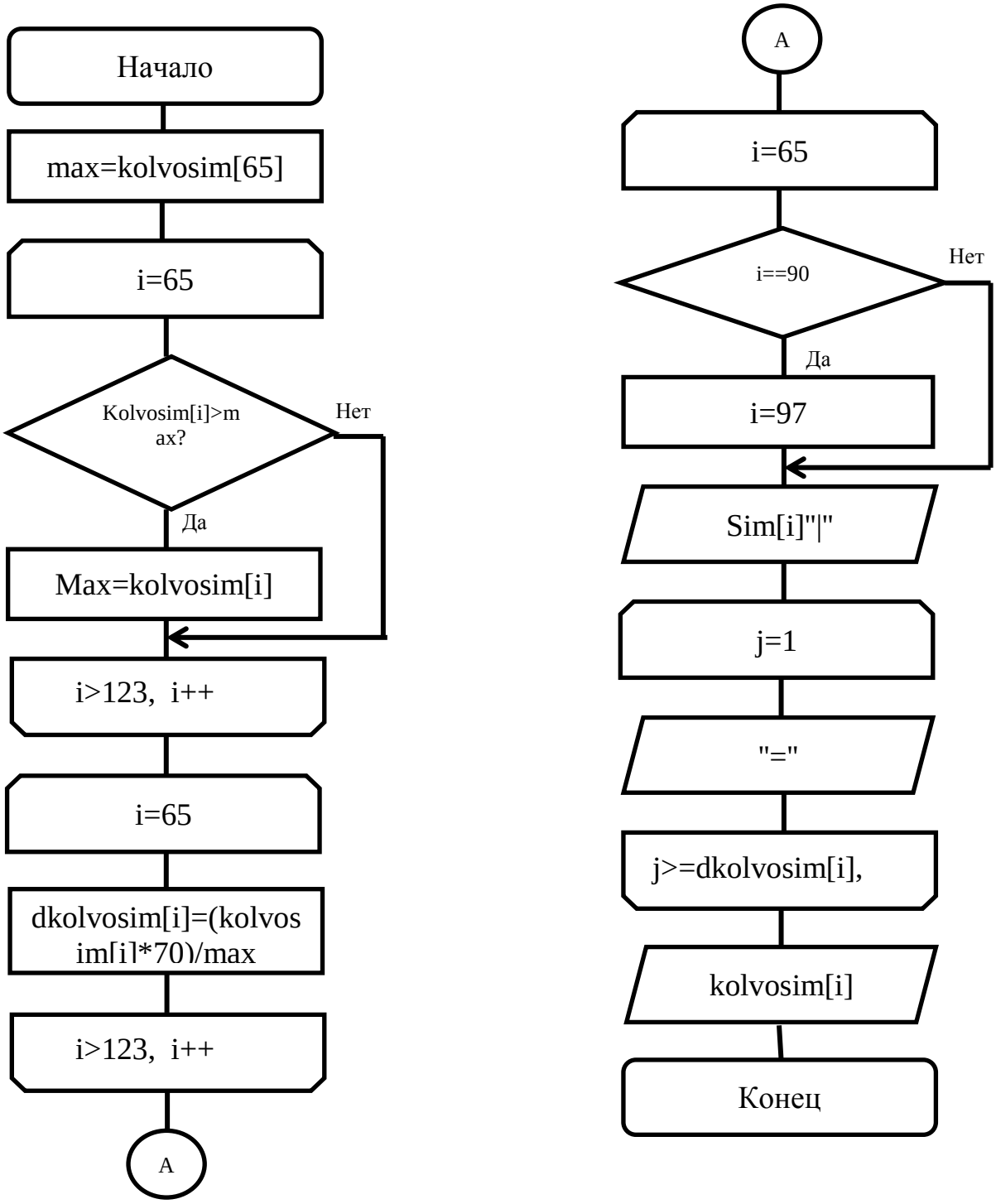


Рисунок А.5 – Блок-схема алгоритма вывода информации об использовании латинских букв в тексте

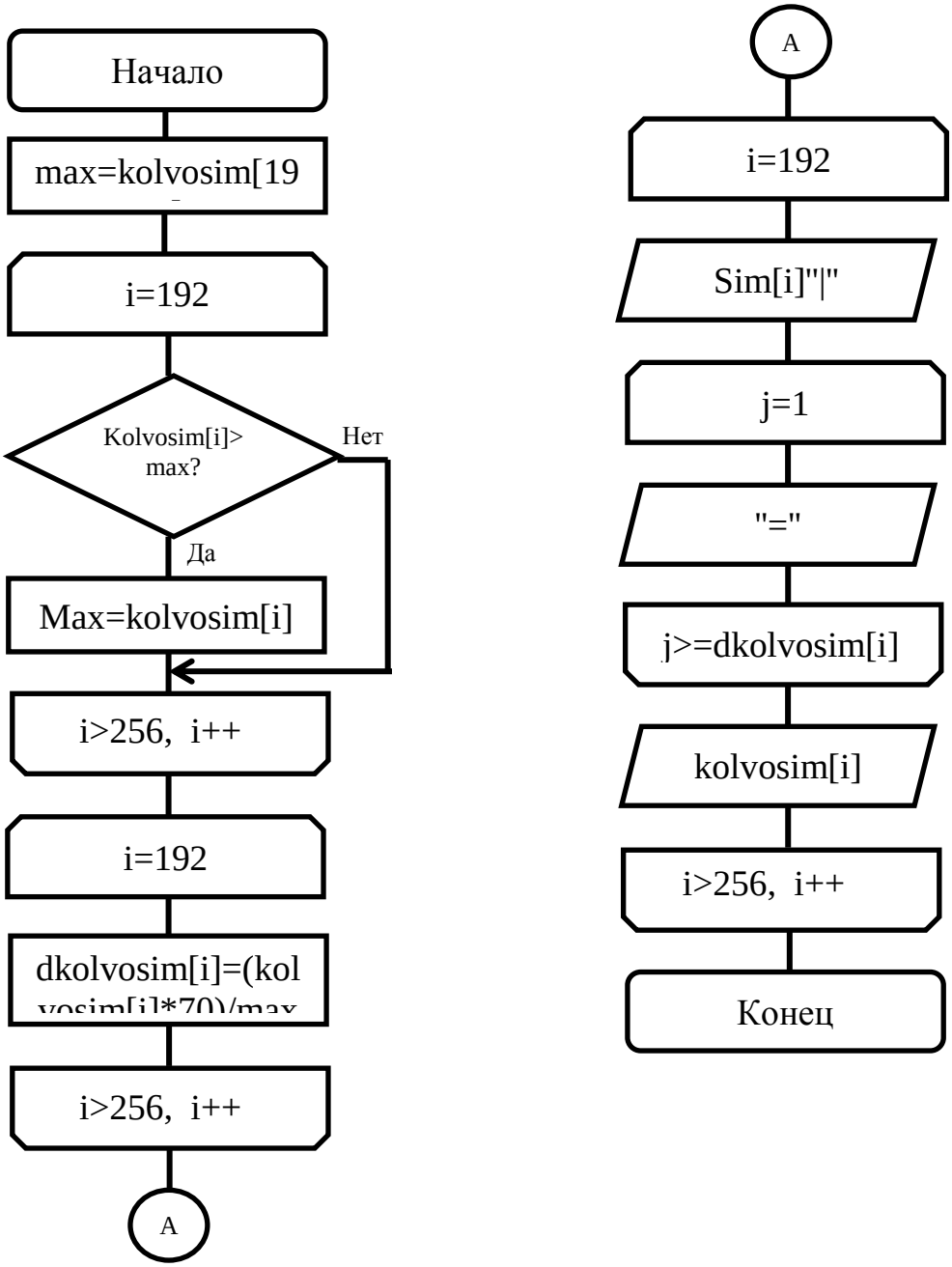


Рисунок А.6 – Блок-схема алгоритма вывода информации об использовании русских латинских букв в тексте

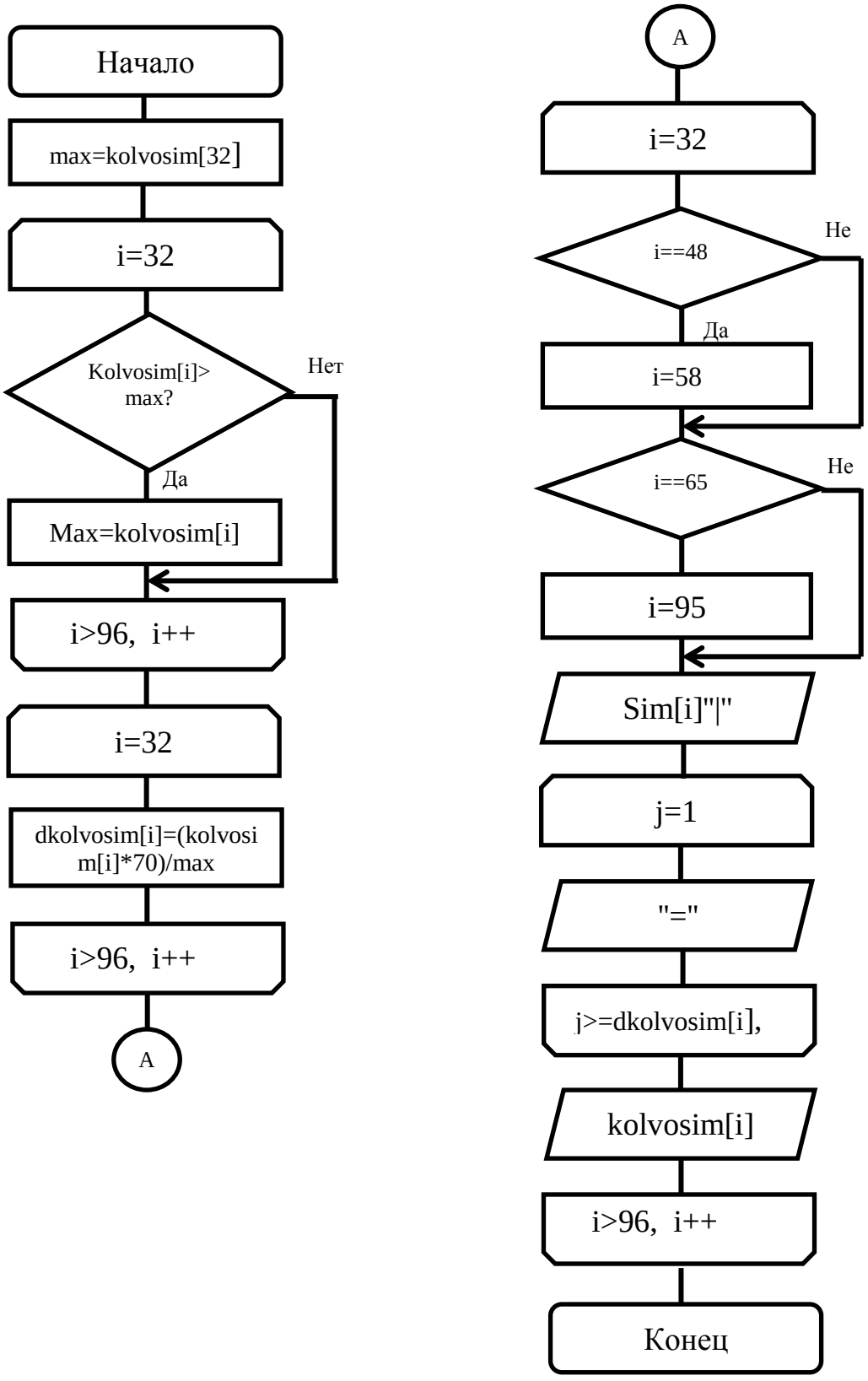


Рисунок А.7 – Блок-схема алгоритма вывода статистики использования знаков препинания

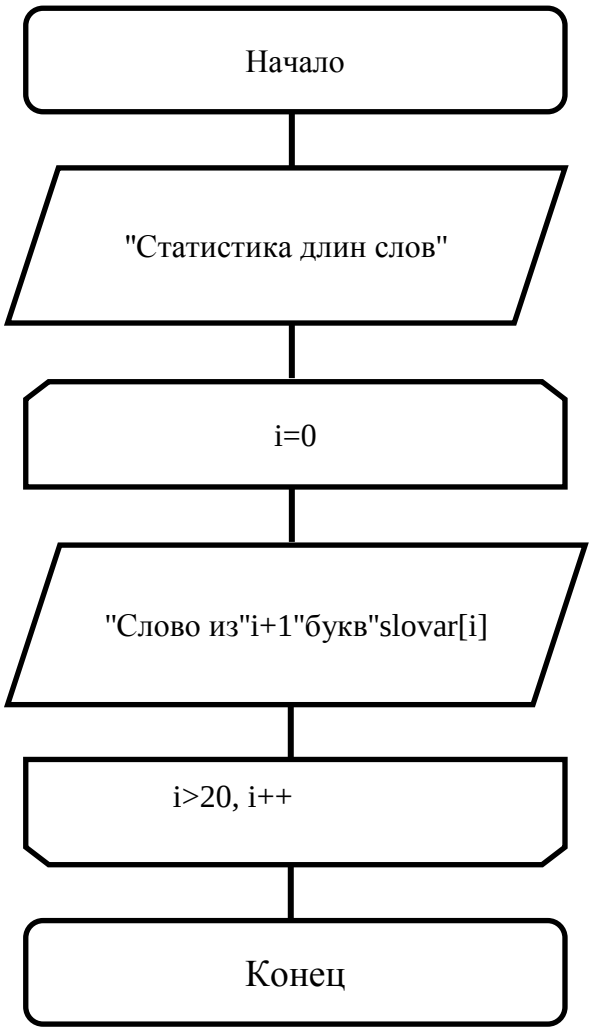


Рисунок А.8 - Блок-схема алгоритма вывода частоты употребления слов разной длины