*Министерство образования и науки*

*Республики Казахстан*

*Петропавловский колледж*

*железнодорожного транспорта*

*СПЕЦИАЛЬНОСТЬ*

*Вычислительная техника*

*и программное обеспечение*

*КУРСОВОЙ ПРОЕКТ*

*на тему: «Преобразование строки по принципу словаря»*

*проверил: выполнил  
Иванов И.Ю. Петров И.Н.  
 гр. ПО-31*

# Задание на курсовое проектирование

Разработать и составить алгоритмы - при вводе слов, программа должна автоматически по алфавиту выставлять список слов и затем на языке программирования C++ составить программу, которая реализует данные алгоритмы.

# Содержание

[1. Задание на курсовое проектирование 2](#_Toc418404169)

[2. Содержание 3](#_Toc418404170)

[3. Введение 4](#_Toc418404171)

[4. Назначение и область применения 5](#_Toc418404172)

[5. Описание алгоритма 6](#_Toc418404173)

[6. Блок-схема программы 7](#_Toc418404174)

[7. Блок-схема алгоритма сортировки строк 8](#_Toc418404175)

[8. Листинг программы, реализующей алгоритмы 10](#_Toc418404176)

[9. Пример работы программы в консольном режиме 14](#_Toc418404177)

[10. Контрольные примеры работы программы 15](#_Toc418404178)

[Исходный текст 15](#_Toc418404179)

[Результат работы программы 15](#_Toc418404180)

[11. Заключение 19](#_Toc418404181)

# Введение

Проблема сортировки больших массивов строк заключается в том, что имеющиеся оценки алгоритмов сортировки расчитываются исходя их среднего числа попарных перестановок элементов. Так для алгоритма Быстрой сортировки имеется оценки , где – число элементов массива.

В тоже время при сравнении двух символьных строк длины требуется провести до операций попарного сравнения символов строк.

Таким образом фактически, для алгоритма Быстрой сортировки строк сиволов, оценка будет , где – средняя длина строк в массиве.

Составление более производительного алгоритма сортировки требует учитывать свойства сортируемых элементов. Так для строк этим свойством является возможность проводить сравнение их не только в лексекографическом порядке, то также сравнение их на основе кодов символов в заданной позиции от начала строки.

# Назначение и область применения

Программа может использоватся для составления писков слов, используемых в различны статьях, или страницах интернет, для дальнейшей индексации источников информации.

# Описание алгоритма

Проблема сортировки больших массивов строк заключается в том, что имеющиеся оценки алгоритмов сортировки расчитываются исходя их среднего числа попарных перестановок элементов. Так для алгоритма Быстрой сортировки имеется оценки , где – число элементов массива.

В тоже время при сравнении двух символьных строк длины требуется провести до операций попарного сравнения символов строк.

Таким образом фактически, для алгоритма Быстрой сортировки строк сиволов, оценка будет , где – средняя длина строк в массиве.

Для сортировки больших массивов строк наиболее оптимальными являются алгоритмы на основе корзинной сортировки, где исходный массив разделяется на корзины сперва по первому сиволу, потом каждая корзина разделяется на подкорзины по второму символу и так далее.

Если корзины сформированы так, что они уже являются упорядоченными множествами, то очевидно, что оценка вычислительной трудоёмкости такого алгоритма составит , где – число элементов массива, – средняя длина строк в массиве.

В данной реализации разнесение строк по корзинам осуществляется на основе индекса в строке заданного в программе алфавита.

Очевидно, что разнесение строк по такому признаку сразу сортирует строки, поскольку корзины упорядочены по номера.

Строки с символом, не содержащимся в заданном алфавите, обрабатываются позднее и сортируются исходя из кода символа.

# Блок-схема программы

начало

Чтение слов

ввод

Вывод слов

вывод

конец

Алгоритм сортировки строк

Несортированный массив

Сортированный массив

# Блок-схема алгоритма сортировки строк

начало

Создать массив buckets [B0, B1, B2, …] из массивов строк, размера мощности алфавита

Цикл по элементам входного массива строк

Получить код index символа в позиции level строки

Код символа соответствует концу строки?

Добавить строку в конец выходного массива

Добавить строку в конец корзины bucket[index]

конец цикла

да

нет

Параметры:

Входной массив строк

Выходной массим строк

Позиция текущего символа для сравнения level

конец

Последовательный цикл по элементам массива корзин

конец цикла

Рекурсивный вызов алгоритма сортировки строк для текущей корзины и уровня level+1

В корзине есть строки?

да

нет

# Листинг программы, реализующей алгоритмы

// при вводе слов, программа должна автоматически по алфавиту выставлять список слов.

// ВНИМАНИЕ!!! Данная реализация поддерживает только однобайтовые кодировки символов.

// Для переходя к многобайтовым символам надо внести изменения в функции:

// int code\_of(char ch)

// void string\_sort(std::vector<std::string> &unsorted, std::vector<std::string> &sorted, int level)

#include <algorithm>

#include <cstdio>

#include <iostream>

#include <sstream>

#include <vector>

#include <string>

#include <clocale>

#include <functional>

#include <cctype>

#include <locale>

using namespace std;

// Используемый алфавит

// Определяет очерёдность

char alphabet[] = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZабвгдеёжзийклмнопрстуфхцчшщъыьэюяАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ" ;

// Сортировать с учётом/без учёта регистра символов

enum t\_compare\_mode {

CASEINSENSITIVE = 0,

CASESENSITIVE = 1

};

t\_compare\_mode compare\_mode = CASEINSENSITIVE;

// Удалять/не удалять дубликаты строк

enum t\_duplicate\_mode {

UNIQUE = 0,

DUPLICATE = 1

};

t\_duplicate\_mode duplicate\_mode = UNIQUE;

// Разбивать/не разбивать строки на слова

enum t\_split\_mode {

SPLIT = 0,

NOSPLIT = 1

};

t\_split\_mode split\_mode = SPLIT;

// Разбивать/не разбивать строки на слова

enum t\_trim\_mode {

TRIM = 0,

NOTRIM = 1

};

t\_trim\_mode trim\_mode = TRIM;

// Определение номера корзины по символу

// с использованием алфавита

int index\_of(char ch)

{

//A pointer to the first occurrence of character in str.

//If the character is not found, the function returns a null pointer.

char \* pos = strchr (alphabet, ch);

if(pos == NULL) return -1;

return (int)(pos-alphabet);

}

// Определение номера корзины по символу

// без использования алфавита

int code\_of(char ch)

{

return (int)(unsigned)ch;

}

char \* input\_file\_name = NULL;

char \* output\_file\_name = NULL;

// trim from start

static inline std::string &ltrim(std::string &s) {

s.erase(s.begin(), std::find\_if(s.begin(), s.end(), std::ptr\_fun<int, int>(std::isalpha)));

return s;

}

// trim from end

static inline std::string &rtrim(std::string &s) {

s.erase(std::find\_if(s.rbegin(), s.rend(), std::ptr\_fun<int, int>(std::isalpha)).base(), s.end());

return s;

}

// trim from both ends

static inline std::string &trim(std::string &s) {

return ltrim(rtrim(s));

}

void append\_to(std::vector<std::string> &sorted, std::string s)

{

switch(duplicate\_mode)

{

case UNIQUE:

{

if(sorted.size()==0) sorted.push\_back(s);

else {

std::string last = sorted.back();

if(s.compare(last)!=0) sorted.push\_back(s);

}

}

break;

default:

sorted.push\_back(s);

}

}

// Рекурсивная сортировка строк, основанная на алгоритме корзинной сортировки

void string\_sort(std::vector<std::string> &unsorted, std::vector<std::string> &sorted, int level)

{

int n = strlen(alphabet);

std::vector<std::vector<std::string>> buckets(n+256);

// Разбрасываем несортированную коллекцию строк по корзинам

// Номер корзины уже сразу даёт предварительную сортировку

for(int i=unsorted.size(); i--; )

{

std::string s = unsorted.back();

unsorted.pop\_back();

char ch = s[level];

if(ch == NULL) {

append\_to(sorted,s);

continue;

}

int index = index\_of(ch);

if(index==-1) index=n+code\_of(ch);

buckets[index].push\_back(s);

}

// Применяем рекурсивно ко всем корзинам кроме последней

for(int i=0; i<n+256; i++)

switch (buckets[i].size())

{

case 0:

break;

case 1:

{

std::string s = buckets[i].back();

buckets[i].pop\_back();

append\_to(sorted,s);

}

break;

default:

string\_sort(buckets[i], sorted, level+1);

break;

}

}

int main(int argc, char\* argv[])

{

// Поддержка кириллицы в консоли Windows

// Функция setlocale() имеет два параметра, первый параметр - тип категории локали, в нашем случае LC\_TYPE - набор символов, второй параметр — значение локали.

// Вместо второго аргумента можно писать "Russian", или оставлять пустые двойные кавычки, тогда набор символов будет такой же как и в ОС.

setlocale(LC\_ALL, "");

for(int i=1; i<argc; i++)

if(strcmp(argv[i],"-help")==0)

{

std::cout << "Преобразование строки по принципу словаря" << std::endl;

std::cout << "сортировка строк, основанная на алгоритме корзинной сортировки" << std::endl;

std::cout << "полиминальная сложность имеет оценку O(N\*L)" << std::endl;

std::cout << "где N - число строк, L - средняя длина строки" << std::endl;

std::cout << "что лучше алгоритма Быстрой сортировки с оценкой O(N\*logN\*L)" << std::endl;

}

else if(strcmp(argv[i],"-cs")==0) compare\_mode = CASESENSITIVE;

else if(strcmp(argv[i],"-cis")==0) compare\_mode = CASEINSENSITIVE;

else if(strcmp(argv[i],"-unique")==0) duplicate\_mode = UNIQUE;

else if(strcmp(argv[i],"-duplicate")==0) duplicate\_mode = DUPLICATE;

else if(strcmp(argv[i],"-split")==0) split\_mode = SPLIT;

else if(strcmp(argv[i],"-nosplit")==0) split\_mode = NOSPLIT;

else if(strcmp(argv[i],"-trim")==0) trim\_mode = TRIM;

else if(strcmp(argv[i],"-notrim")==0) trim\_mode = NOTRIM;

else if(strcmp(argv[i],"-input")==0) input\_file\_name = argv[++i];

else if(strcmp(argv[i],"-output")==0) output\_file\_name = argv[++i];

if(input\_file\_name!=NULL) freopen(input\_file\_name,"r",stdin);

if(output\_file\_name!=NULL) freopen(output\_file\_name,"w",stdout);

std::vector<std::string> unsorted;

std::vector<std::string> sorted;

// Буфер для ввода строк

std::string buffer = "";

std::cout << "Вводите строки. "

<< "По окончание ввода введите Ctrl+Z" << std::endl;

while (std::getline(std::cin, buffer))

{

if (buffer.size() > 0) {

// https://notfaq.wordpress.com/2007/08/04/cc-convert-string-to-upperlower-case/

switch (compare\_mode)

{

case CASEINSENSITIVE:

std::transform(buffer.begin(), buffer.end(), buffer.begin(), ::toupper);

break;

case CASESENSITIVE:

break;

}

switch (split\_mode)

{

case SPLIT:

{

std::istringstream ss(buffer);

std::string word;

while(ss>>word)

{

if(trim\_mode == TRIM) word = trim(word);

if(word!="") unsorted.push\_back(word);

}

}

break;

case NOSPLIT:

{

if(trim\_mode == TRIM) buffer = trim(buffer);

// Добавление элемента в конец вектора

if(buffer!="") unsorted.push\_back(buffer);

}

break;

}

}

}

string\_sort(unsorted, sorted, 0);

// Сохраняем количество элементов вектора

unsigned int vector\_size = sorted.size();

// Вывод заполненного вектора на экран

std::cout << "Ваш словарь ( " << vector\_size << " слов )." << std::endl;

for (int i = 0; i < vector\_size; i++) {

std::cout << sorted[i] << std::endl;

}

getchar();

getchar();

return 0;

}

# Пример работы программы в консольном режиме



# Контрольные примеры работы программы

## Исходный текст

WikiLeaks is a not-for-profit media organisation. Our goal is to bring important news and information to the public. We provide an innovative, secure and anonymous way for sources to leak information to our journalists (our electronic drop box). One of our most important activities is to publish original source material alongside our news stories so readers and historians alike can see evidence of the truth. We are a young organisation that has grown very quickly, relying on a network of dedicated volunteers around the globe. Since 2007, when the organisation was officially launched, WikiLeaks has worked to report on and publish important information. We also develop and adapt technologies to support these activities.

WikiLeaks has sustained and triumphed against legal and political attacks designed to silence our publishing organisation, our journalists and our anonymous sources. The broader principles on which our work is based are the defence of freedom of speech and media publishing, the improvement of our common historical record and the support of the rights of all people to create new history. We derive these principles from the Universal Declaration of Human Rights. In particular, Article 19 inspires the work of our journalists and other volunteers. It states that everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers. We agree, and we seek to uphold this and the other Articles of the Declaration.

## Результат работы программы

Ваш словарь ( 135 слов ).

A

ACTIVITIES

ADAPT

AGAINST

AGREE

ALIKE

ALL

ALONGSIDE

ALSO

AN

AND

ANONYMOUS

ANY

ARE

AROUND

ARTICLE

ARTICLES

ATTACKS

BASED

BOX

BRING

BROADER

CAN

COMMON

CREATE

DECLARATION

DEDICATED

DEFENCE

DERIVE

DESIGNED

DEVELOP

DROP

ELECTRONIC

EVERYONE

EVIDENCE

EXPRESSION

FOR

FREEDOM

FROM

FRONTIERS

GLOBE

GOAL

GROWN

HAS

HISTORIANS

HISTORICAL

HISTORY

HOLD

HUMAN

IDEAS

IMPART

IMPORTANT

IMPROVEMENT

IN

INCLUDES

INFORMATION

INNOVATIVE

INSPIRES

INTERFERENCE

IS

IT

JOURNALISTS

LAUNCHED

LEAK

LEGAL

MATERIAL

MEDIA

MOST

NETWORK

NEW

NEWS

NOT-FOR-PROFIT

OF

OFFICIALLY

ON

ONE

OPINION

OPINIONS

ORGANISATION

ORIGINAL

OTHER

OUR

PARTICULAR

PEOPLE

POLITICAL

PRINCIPLES

PROVIDE

PUBLIC

PUBLISH

PUBLISHING

QUICKLY

READERS

RECEIVE

RECORD

REGARDLESS

RELYING

REPORT

RIGHT

RIGHTS

SECURE

SEE

SEEK

SILENCE

SINCE

SO

SOURCE

SOURCES

SPEECH

STATES

STORIES

SUPPORT

SUSTAINED

TECHNOLOGIES

THAT

THE

THESE

THIS

THROUGH

TO

TRIUMPHED

TRUTH

UNIVERSAL

UPHOLD

VERY

VOLUNTEERS

WAS

WAY

WE

WHEN

WHICH

WIKILEAKS

WITHOUT

WORK

WORKED

YOUNG

# Заключение

Изучен алгоритм корзинной сортировки

Изучена стандартная библиотека классов std в C++