|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  **САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  **ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  **«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)**  **Кафедра РЭС** |
| **ОТЧЕТ по лабораторной работе № по дисциплине «Информатика» Тема: Разработка консольного приложения для обработки и экспорта телефонной книги.** |
| |  |  | | --- | --- | | Студент гр. 1183 | Чаминов Д. А. | | Преподаватель | Ситников И. Ю. | |

# Содержание

[Содержание 2](#_Toc86833440)

[Спецификация задания 3](#_Toc86833441)

[Описание алгоритма 4](#_Toc86833442)

[Блок-схема алгоритма 5](#_Toc86833443)

[Выбор и обоснование типов переменных 8](#_Toc86833444)

[Вводимые и выводимые параметры и их типы 9](#_Toc86833445)

[Структура проекта 10](#_Toc86833446)

[Текст программы 11](#_Toc86833447)

[Копии экрана при работе программы 12](#_Toc86833448)

[Контрольный пример, сравнение с эталоном 13](#_Toc86833449)

[Результаты 14](#_Toc86833450)

# Спецификация задания

Разрабатываемая программа должна

* Разрабатываться на языке C++ в среде MS Visual Studio
* Работать на операционных системах MS Windows (7 и выше)
* Поддерживать кириллицу
* Производить следующие операции по запросу пользователя:
  + Ввод данных контактов при помощи клавиатуры
  + Ввод данных контактов из файла
  + Импорт данных контактов из файла, использующего Google csv формат записи контактов
  + Поиск контакта
  + Изменение имени контакта
  + Изменение номера контакта
  + Удаление контакта из телефонной книги
  + Вывод данных контактов на экран в виде таблицы
  + Вывод данных контактов в виде таблицы в файл
  + Экспорт контактов в файл в формате Google csv
* Использовать двоичный поиск
* Сортировать данные в телефонной книге по алфавиту
* Включать в себя по крайней мере 2 пользовательских класса

# Описание алгоритма

При старте программа предлагает пользователю выбрать действие.

При выборе действия «Ввести с клавиатуры», программа должна добавлять записи, создаваемые по вводимым пользователем данным, в телефонную книгу, пока в качестве номера телефона не будет введен «0». Затем пользователь должен подтвердить окончание ввода, в случае если пользователь отказывается, ввод продолжается.

При выборе действия «Ввести из файла», программа должна предложить пользователю ввести имя файла. В случае если файл не удается открыть, программа должна предложить пользователю вернуться к выбору действия, открыть другой файл или же завершить выполнение программы. В случае если файл был открыт успешно, программа должна интерпретировать каждые 2 строки этого файла, как пару – имя контакта и его номер.

При выборе действия «Импортировать (Google csv)», программа для открытия файла повторяет действия, указанные выше, а затем, конвертирует данные, которые в файле разделяются запятыми, в записи телефонной книги. Этот формат подразумевает наличие заголовка с перечислением полей, а затем в каждой строчке – данные контакта. Для получения заголовка и этих данных программа проверяет каждый символ строки на соответствие запятой, кавычкам или символу новой строки. Если текущий символ – запятая, а кавычек в ходе исследования текущего поля не встречалось, то следующий символ принадлежит следующему полю. Если текущий символ – символ перевода каретки, то это последнее поле контакта. Для интерпретации данных в качестве контакта телефонной книги программа ищет среди полей заголовка поле «Name» и поле «Phone 1 - Value». В случае, если для контакта было определено несколько номеров, поле телефонного номера содержит «:::», разделяющее телефонные номера. Программа выделяет из поля номера номера (если требуется) и для каждого из номеров создает запись в телефонной книге, добавляя в имя контакта порядковый номер этого телефонного номера.

Примечание: при импорте программа не сохраняет и не использует дополнительные (не имя или номер) поля контактов.

При выборе действия «Найти номер», в случае если телефонная книга пуста, пользователь получает соответствующее сообщение и возвращается к выбору действия. В ином случае программа получает от пользователя имя контакта. После этого программа начинает поиск контакта по имени: по результатам сравнения искомого имени с серединой отрезка, в котором происходит поиск (сравнение происходит без учета регистра), граница этого отрезка (правая, если искомый элемент расположен раньше по алфавиту или элементы совпадают, и левая в ином случае) устанавливается в середину. Сдвиг границ поиска происходит по этому алгоритму пока не сомкнутся. Тогда правая граница и будет искомым элементом, или ближайшим слева (лексикографически) к искомому элементом. Этот же алгоритм поиска используется при вставке записи в телефонную книгу, обеспечивая таким образом алфавитный порядок записей. После этого программа предлагает пользователю выбор – вернуться к выбору действий, изменить имя контакта, изменить телефонный номер или удалить контакт. Затем в соответствии с выбором пользователя программа изменяет соответствующее поле записи, возвращает пользователя к выбору действия или удаляет контакт.

При выборе действия «Вывести на экран», программа для каждой записи получает имя и номер, вычисляет наибольшую длину имени (сравнивая текущий максимум длины с длиной текущего имени); добавляет справа к строке для имени символы разделителей, пока длина строки меньше максимальной длины имени, и 3 разделительных символа; добавляет разделительные символы к строке номера справа, пока длина номера не достигнет 20 символов; аналогично увеличивает длину строки, содержащей, номер записи в телефонной книге до 4х символов (если требуется); выводит полученные строки на экран в следующем порядке: номер записи, имя контакта, телефонный номер контакта.

При выборе действия «Вывести в файл» программа получает имя файла, в случае если файла с таким именем не существует, он будет создан, но если создание или открытие файла не удалось, то пользователю будут предложены те же действия, что и при неудачном открытии файла при выборе «Ввести с клавиатуры». Затем, таблицу, сформированную по тому же алгоритму, что и для пункта «Вывести на экран» программа записывает в указанный файл. (Если файл с таким именем не существовал, он будет создан)

При выборе действия «Экспортировать (Google csv)» программа открывает файл как при действии «Вывести в файл». В файл в качестве заголовка записываются имена полей имени и номера, а затем для каждой записи в телефонной книге – строковые представления соответствующих полей.

# Блок-схема алгоритма

начало

Phonebook pb

конец

recordFind(pb)

нет

да

action == 7

fileOutput(pb)

export(pb)

нет

да

action == 6

нет

да

action == 5

action == 4

pb.print(0,std::cout,’.’)

да

нет

нет

да

нет

да

нет

да

size\_t action = choose\_list(8, “Ввести с клавиатуры\0Ввести из файла\0Импортировать (Google csv)\0Вывести на экран\0Вывести в файл\0Экспортировать (Google csv)\0Найти номер\0Выйти”);

import(pb)

action == 3

inputFromFile(pb)

action == 2

action == 1

pb.input()

Алгоритм inputFromFile:

Алгоритм Phonebook::inputFromFile:

начало

конец

ifstream fin(filename);  
pb.inputFromFile(fin);  
fin.close();

ввод filename

получить Phonebook& pb

нет

да

fin.good()

PhoneRecord r;

r.input(fin);

addRecord(r);

конец

получить std::istream fin

начало

Алгоритм PhoneRecord::input:

начало

конец

setName(name);

setNumber(number);

std::string name, number;

Получить std::istream & input

ввод name, number

Алгоритм PhoneRecord::setName:

Конец

получить std::string name

Начало

this->name=name

Алгоритм PhoneRecord::setNumber:

нет

да

нет

да

начало

конец

number.plus = (s[0]==’+’);

number.num = std::stoll(s);

s.size()==0

number.num=0;

number.plus=0;

s=filter(s)

filterstr

получение std::string s, bool filterstr=true

Алгоритм PhoneRecord::filter:

нет

нет

да

да

конец

вернуть result

i = i+1;

is\_first=false;

result=result+s[i];

(s[i]>=’0’ И s[i]<=’9’) ИЛИ (is\_first И s[i]==’+’)

i<n

bool is\_first=true;

size\_t n = s.size();

std::string result=””;

size\_t i=0;

получение const std::string& s

начало

Алгоритм Phonebook::addRecord:

конец

phbook.emplace(i, r);

std::list<PhoneRecord>::iterator i = findRecord(r);

получить PhoneRecord r

начало

Алгоритм Phonebook:: FindRecord:

нет

да

да

нет

да

нет

m=(left+right)/2;

конец

right=m;

left=m;

phbook[m]>=r

вернуть phbook.begin()+1+left

phbook.size() == 0

вернуть phbook.begin()

size\_t left=-1,

right = phbook.size();

size\_t m=(left+right)/2;

right-left>1

начало

получить const PhonrRecord& r

Алгоритм import:

конец

std::ifstream fin(filename);  
pb.importPhonebook(fin);

fin.close();

ввод std::strring filename

получить Phonebook& pb

начало

Алгоритм Phonebook::importPhonebook:

нет

нет

да

да

std::string buf;  
getline(fin,buf);

std::vector<std::string> f;

size\_t start\_i=0;

получить std::ifstream fin

начало

конец

fin.good()

addCSVRecord(fin,f);

std::pair<std::string,size\_t> tmp = getCSVField(buf,start\_i);;

f.push\_back(tmp.first);

start\_i=tmp.second;

buf.size()>start\_i

Алгоритм Phonebook::addCVSRecord:

нет

нет

нет

да

да

да

конец

addRecord(name,number)

name = f[i];

i++

number = f[i];

f [i] == ” Phone 1 - Value”

f [i] == ”Name”

i<f.size()

std::string name, number;

size\_t i=0;

std::pair<std::string,size\_t> tmp=getCSVField(buf,start\_i);

v.push\_back(tmp.first);

start\_i=tmp.second;

buf.size()>start\_i

std::string buf;

size\_t start\_i=0;

std::vector<std::string> v;

getline(in,buf);

получить std::istream in, const std::vector<std::string>& f

начало

Алгоритм getCSVField:

нет

нет

да

да

конец

вернуть std::pair<std::string, size\_t>(r,i+)

i++

q = НЕ q

r=r+s[i]

s[i]==’ ” ’

(i<s.size()) И (q ИЛИ (s[i]==’,’))

std::string r =””;

bool q=false;

начало

получить std::string s, size\_t i

Алгоритм fileOutput:Алгоритм Phonebook::print:

конец

std::ofstream fout(filename);

pb.print(0,fout,’.’);

fout.close();

ввод std::string filename

получить const Phonebook& pb

начало

нет

да

конец

(\*i).print(name\_length, output, filler)

output: inflate\_string(std::to\_string(n+1),4)

size\_t n =0;

std::list<PhoneRecord>::iterator i = phbook.begin()

i == phbook.end()

начало

Получить size\_t name\_length, std::ostream & output, char filler

Алгоритм PhoneRecord::print:

начало

std::string name = inflate\_string(nickname, name\_length, StrAlign::left, filler);

std::string n = inflate\_string(getNumber(), 20, StrAlign::right, filler);

output: name, n

конец

Получить size\_t name\_length, std::ostream & output, char filler

Алгоритм PhoneRecord::getNumber:

нет

да

конец

вернуть ‘+’ + std::to\_string(number.num)

вернуть std::to\_string(number.num)

number.plus

начало

Алгоритм inflate\_string:

вернуть s

конец

нет

нет

нет

нет

нет

нет

нет

да

да

да

да

да

да

да

s = inflate\_string(s, (w - n) / 2 + n, StrAlign::left, filler);

s = inflate\_string(s, (w - n) / 2 + (w - n) % 2 + s.size(), StrAlign::right, filler);

s.erase(0, (n - w) / 2);

s.erase(w, (n - w) / 2 + (n - w) % 2);

n>w

n>w

n<w

s.erase(w,n-w);

s=s+filler;

n++;

a==left

s.erase(0,n-w);

n>w

s=filler+s;

n++;

n<w

a==right

size\_t n = s.size();

получить std::string s, size\_t w, StrAlign:: a, char filler

начало

Алгоритм Phonebook::export:

Алгоритм export():Алгоритм recordFind:

конец

i++

out<<phbook[i].getName();  
out<<phbook.getNumber();

i<phbook.size()

size\_t i=0

out<<”Name, Phone 1 – value”

получить std::ostream out;

начало

std::ofstream fout(filename);

pb.export(fout);

fout.close()

конец

ввод std::string filename

получить const Phonebook& pb

начало

rp->setNumber( number)

ввод std::string number

rp->setName(name)

action == 3

rp->setName(name)

ввод std::string name

action == 2

конец

action == 1

ввод size\_t action

ввод std::string name

auto rp=pb.findRecord(PhoneRecord(name,”0”,false))

начало

получить const Phonebook& pb

ЕЩЕ найти запись, экспорт

Блок-схема нужна для представления алгоритма в виде последовательности операций и переходов между ними.

Основные вычислительные алгоритмы содержат повторяющиеся действия, при этом повторение не должно быть бесконечным – тогда программа никогда бы не заканчивалась и не позволяла получить результат. Существуют и длительно выполняющиеся программы - например программы игр или операционных систем, которые работают все время пока их не выключают. Это алгоритмы другого типа, по сути, не являющиеся вычислительными. В нашем курсе мы будем писать программы конечными, предназначенными для решения конкретной задачи, и когда результат получен, вычислительный алгоритм тоже заканчивается.

В наших блок-схемах появляются блоки НАЧАЛО и ЗАВЕРШЕНИЕ, по правилам оформления блок-схем [] это прямоугольники со скругленными углами. У НАЧАЛА один выход и ни одного входа, у ЗАВЕРШЕНИЯ – один вход и ни одного выхода. Другие блоки содержат один вход и как минимум один выход. Если у блока нет ни одного выхода, он «висит в воздухе», то программа, попав в этот блок останется там навсегда – проверяйте ваши схемы на наличие блоков, висящих в воздухе.

Блоки соединяются линиями, показывающими последовательность действий. Поэтому линия, соединяющая блоки, должна быть со стрелкой, показывающей откуда и куда переходит действие.

Для начала вычислений нужно ввести исходные данные, а в конце работы напечатать результат. Блоки ввода и вывода изображаются параллелограммами.

Наконец, в алгоритме бывают ветвления, то есть выполнение действий может идти, например по двум путям (простое условие). Блок проверки условия содержит описание проверки (что проверяется) и два выхода, которые подписываются «ДА» и «НЕТ» и соответствуют результату проверки.

# Выбор и обоснование типов переменных

В программе вычисления выполняются над переменными - числами, записанными в памяти компьютера, называемыми операндами. Например, С=А+В – простое выражение, которое означает взять число из ячейки А сложить его с числом из ячейки В и записать результат в ячейку С. В этой записи, несмотря на ее простоту, есть неопределенность для программы, заключающаяся в том, что числа в программе могут быть представлены в различных форматах, проще говоря занимать разное по размеру место в памяти машины, и это может существенно влиять на результат. В дальнейшем мы рассмотрим детально эти представления и правила преобразования типов, однако отчет должен точно описывать используемые в программе типы переменных для правильной работы с обоснованием использования выбранного типа. Например А, В. С – имеют целочисленный тип со знаком int.

# Вводимые и выводимые параметры и их типы

В этом разделе приводится полный перечень данных, которые программа получает, от оператора, в нашем случае, и через ввод с клавиатуры. Вывод результатов также должен быть описан, какие условные обозначения, сообщения и другую вспомогательную информацию может выдавать программа с необходимыми пояснениями. Например: в программе вводятся слагаемые А и В, оба представляют собой целые числа со знаком, не превышающие по модулю значения 200000000. Результат расчета выводится в строке «А+В=» и программа ожидает нажатия клавиши оператором для завершения.

# Структура проекта

Программа разрабатывается в современной среде разработки Microsoft (r) Visual Studio. Это очень хороший набор программ и вспомогательных средств для профессионального программирования. Процесс написания программы – это написание текста на одном из языков программирования, мы будем использовать языки С и С++. Сам текст является только средством для облегчения восприятия человеком последовательности машинных команд, которые «видит» машина в программе. Поэтому текст с языка С должен быть «переведен» на язык машинных команд, и поскольку этот перевод должен быть не вольным, а точным, процесс называется компиляцией, а выполняет его специальная программа – компилятор. Для нас важно, что тексты в программе – это текстовые файлы на языке С или С++, их имя может быть любым (избегайте использования кириллицы), а расширение .cpp для текста С++ или для классического С .c. Кроме файлов текста, которые обычно содержат тексты выполняемых операций в проекте есть файлы заголовков, которые в основном являются пояснениями компилятору, какие программист использует определения в своих текстах. Файлы заголовков имеют расширение .h от английского headers. Ну, и раз текстов уже несколько, должен быть файл проекта, который содержит в себе как минимум перечень файлов текста, файлов заголовков и еще инструкции в какую форму наш проект должен исходные файлы собрать. Расширение этого файла проекта .vcxproj от Visual C Extended Project. Этап сборки скомпилированных файлов и приводит в случае успеха к созданию исполняемого файла, с расширением .exe (executable). Исполняемый файл не универсален, он строится с учетом особенностей операционной системы и как правило не работает в другой ОС.

# Текст программы

//мой код:

# Копии экрана при работе программы

\*скриншоты (лучше win+shift+s в режиме окна)\*

# Контрольный пример, сравнение с эталоном

Ну тут типа информация по прохождению теста / тестов

# Результаты

К результатам работы относятся характеристики разработанной программы, такие, как: • тип ОС, для которой разработано приложение • тип приложения (консольное, оконное…) • путь, имя и размер исполняемого .exe файла • ограничения на исходные данные • результаты проверки выходных значений по независимому источнику • достоинства и недостатки программы