|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  **САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  **ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  **«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)**  **Кафедра РЭС** |
| **ОТЧЕТ по лабораторной работе №6  по дисциплине «Информатика» Тема: Использование классов для создания 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++,
* разрабатываться в среде MS Visual Studio,
* работать в операционных системах MS Windows 7 и выше,
* использовать хотя бы 2 пользовательских класса,
* принимать данные контактов телефонной книги (имя и номер), сортировать их и выводить в виде таблицы с выравниванием.

# Описание алгоритма

Для хранения телефонной книги создается экземпляр класса Phonebook, который содержит всего одно поле типа std::vector<PhoneRecord>. В этом классе определены функция бинарного поиска по записям в книге (методом сдвига границ поиска по результатам сравнения записи в середине отрезка с искомой), функции добавлений записей в книгу, которые при помощи функции двоичного поиска находят место для вставки (реализуя таким образом сортировку вставками), функции ввода телефонной книги из потока ввода, использующие функцию добавления записи, пока не считают терминальный элемент (нулевой номер), функция распечатки, которая вызывает распечатку для каждой записи в векторе.

Класс PhoneRecord содержит член – структуру, содержащую два поля – телефонный номер (выделено 50 бит, так как этого хватит для представления 15-значного числа) и индикатор того, начинается ли номер с символа ‘+’. Для этого класса определены дружественная функция сравнения int8\_t compair, которая сравнивает строки имен двух контактов без учета регистра, функция фильтрации строки телефонного номера, которая проверяет каждый символ строки на то, является он ведущим плюсом или цифрой, «get’теры» и «set’теры» для имени и номера телефона (так как они находятся в поле спецификатора доступа protected), конструктор по умолчанию и инициализирующий конструктор, функция ввода записи из потока ввода, функция вывода записи в поток вывода, использующая функцию расширения строки inflate\_string.

inflate\_string добавляет указанные символы к строке так, чтобы исходная строка выравнивалась либо по левому, либо по правому краю, любо по центру и занимала указанный объем.

При старте программа создает телефонную книгу, вызывает функцию ввода через стандартный поток ввода из консоли, затем вызывает функцию вывода телефонной книги в стандартный поток вывода на консоль.

# Блок-схема алгоритма

Конец

system(“pause”);

pb.input(“0”);

pb.print(50,std::cout,’.’)

Phonebook pb;

Начало

Алгоритм Phonebook::input:

да

нет

нет

да

конец

начало

r->input(input, output, prompt\_name. prompt\_number)

PhoneRecord r;

Получить std::string terminator, std::istream & input, std::ostream & output, std::string prompt\_name, std::string prompt\_number

did\_user\_accept()

r->getName() == terminator

addRecord(r)

Алгоритм PhoneRecord::input:

начало

конец

setName(name);

setNumber(number);

std::string name, number;

Получить std::string terminator, std::istream & input, std::ostream & output, std::string prompt\_name, std::string prompt\_number

output: prompt\_name

input: name

output: prompt\_number

input: 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

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 Phonebook::print:

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

# Выбор и обоснование типов переменных

Программа использует пользовательский тип Phonebook – класс определяющий работу с телефонной книгой. Внутри него содержится контейнер std::vector<PhoneRecord>, являющий автоматическим динамическим массивом. Случайный доступ к элементам позволяет использовать двоичный поиск. В качестве полей имени и для строкового представления номера используется тип std::string (std::basic\_string<char>), так как это позволяет более удобно вводить и выводить данные, а также обеспечивает автоматический контроль за памятью. Для хранения телефонного номера используется анонимная структура, содержащая два поля uint64\_t num : 50 (для хранения цифр номера телефона, 50 бит обеспечивает представление 15-значного числа), и uint64\_t plus : 1 (индикатор того, начинается ли номер с плюса).

# Вводимые и выводимые параметры и их типы

На ввод программе через стандартный поток ввода консоли после приглашения ко вводу подаются имена контактов и телефонные номера. Для окончания ввода необходимо в качестве телефонного номера ввести «0» или строку, не содержащую цифр или знака «+». Эти данные записываются в переменные типа std::string. После завершения ввода необходимо ввести «y», если вы подтверждаете окончание ввода, и «n» если вы хотите продолжить ввод.

После подтверждения окончания ввода начнется вывод записей в виде таблицы. В качестве шапки таблицы выводятся char[25], size\_t, char[11], а также строки, длины которых задаются параметрически. Затем выводятся строки таблицы (состоящие из номера записи size\_t, имени контакта std::string и номера телефона в строковом представлении std::string), длину выводимого имени можно задать, в программе она равна 40 символов. Имя выравнивается по левому краю. После имени в 20 символов с выравниванием по правому краю выводится телефонный номер.

# Структура проекта

Файлы проекта располагаются в двух директориях.

Labs\Lab6:

* 24.11.2021 14:04 409 Lab6.cpp
* 22.11.2021 15:37 7 326 Lab6.vcxproj
* 22.11.2021 15:37 1 318 Lab6.vcxproj.filters
* 22.11.2021 15:23 168 Lab6.vcxproj.user

Labs\libs:

* 28.11.2021 18:27 9 203 PhoneNumerCPP.cpp
* 28.11.2021 00:51 7 999 PhoneNumerCPP.h

# 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