|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  **САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  **ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  **«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)**  **Кафедра РЭС** |
| **ОТЧЕТ по лабораторной работе №4  по дисциплине «Информатика» Тема: Использование си-структур 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 13](#_Toc86833444)

[Вводимые и выводимые параметры и их типы 14](#_Toc86833445)

[Структура проекта 15](#_Toc86833446)

[Текст программы 16](#_Toc86833447)

[Копии экрана при работе программы 17](#_Toc86833448)

[Контрольный пример, сравнение с эталоном 18](#_Toc86833449)

[Результаты 19](#_Toc86833450)

# Спецификация задания

Разрабатываемая программа должна:

* Разрабатываться в среде MS Visual Studio
* Работать на операционных системах MS Windows (7 и выше)
* Быть написана на языке C (C++)
* Принимать данные о записях в телефонной книге, выводить телефонную книгу в виде таблицы, содержащей номер записи, имя контакта и телефонный номер.

# Описание алгоритма

На вход программе подаются данные контактов телефонной книги. Программа считывает их, пока в качестве телефонного номера не будет введен «0».

Строки, содержащие имя и номер, не должны быть длиннее 64 символов каждая, а общее количество записей в телефонной книге не должно превышать 100.

Для каждого контакта программа динамически создает запись MyPhoneRecord при помощи функции MyPhoneRecord\* create\_record(constchar\* nick, const char\* number) и заносит ее в массив. Для этого дополнительно в динамической памяти выделяется место под имя, строка, содержащая номер, фильтруется (каждый символ, не являющийся ведущим плюсом или цифрой игнорируется), телефонный номер записывается в структуру MyPhoneNumber, содержащей два поля: unsigned \_\_int64 number:50 (50 бит хватит для представления 15значного положительного целого числа) и unsigned \_\_int64 plus:1 – индикатор того был ли в начале номера «+».

После окончания ввода программа для каждой записи вызывает функцию void print\_record(constMyPhoneRecord\* r, int nick\_len, char filler=’ ‘), которая форматирует строки имени и номера при помощи функции char\* inflate\_string(char\* s, unsigned int width, StringAlign align, char filler), и выводит их в виде строки таблицы, также в ходе выполнения программы выводится номер строки.

# Блок-схема алгоритма

Начало

да

нет

delete\_record(phonebook[j])

Начало

j<i

нет

print\_record(phonebook[j])

Вывод 1+j

int j=0;

char name\_buf[64];

char num\_buf[64];

да

did\_user\_accept()

phonebook[i] = createRecord (name\_buf, num\_buf)

i++;

!strcmp ( name\_buf, “0”)

Ввод name\_buf, num\_buf

MyPhoneRecord\* phonebook[100];

int i=0;

Алгоритм createRecord:

Алгоритм filter\_string:

Конец

Освободить filtered\_num

Вернуть res

res->number.num=atoll(filtered\_num)

нет

нет

да

да

filtered\_num[0]=’+’

res->number.plus=1

res->number.plus=0

filtered\_num[0] == ’-’

filtered\_num[0] == ’+’

strcpy(filterednum, number)  
filter\_string(filtered\_num)

Выделение памяти под char filtered\_num[]

strcpy(res->nickname, nick)

Выделение динамической памяти для MyPhoneRec res и res->nickname

Получение char\* nick, char\* number

Начало

нет

да

нет

да

конец

Вернуть s

s[i]==’\0’

i=i+1

s[j]=s[i]

j=j+1

(s[i]>=’0’ И s[i]<=9) ИЛИ ((s[i]==’+’ ИЛИ s[i]==’-‘) И j==0)

int i=0, j=0;

Получение char\* s

начало

Алгоритм print\_record:

Алгоритм inflate\_string :

нет

да

да

Конец

Освободить nick

Вывод nick, number\_reverse

inflate\_string(number\_reverse, 20, StringAlign::right, filler)

reverse\_string(number\_reverse)

number\_reverse[tmp]=’\0’

tmp = tmp + 1

number\_reverse[tmp]=’\0’

r->number.plus

n==0

нет

number\_reverse[tmp] = n%10 +’0’

tmp = tmp +1

n = n/10

char num\_reverse[21]  
unsigned \_\_int64 n =   
r->number.num  
int tmp = 0

inflate\_string(nick, nicklen, StringAlign::right, filler)

nick[nick\_len]=’\0’

strncnpy(nick, r->nickname, nicklen)

Выделение памяти под char nick[nicklen+1]

Получение MyPhoneRecord\* r,

int nick\_len, char filler

Начало

нет

да

нет

нет

да

да

нет

нет

да

да

конец

s[w]=’\0’

inflate\_string(s, (w - n) / 2 + n, StringAlign::right, filler)

inflate\_string(s, w, StringAlign::left, filler)

w-i<=n

s[i]=s[n-w+i]

s[i]=filler

i = i+1

int i = n

i<w

s[i]=filler

i=i+1

i<w

int i = n

align == right

align == left

unsigned int n=strlen(s)

Получение char\* s, unsigned int w, StringAlign align, char filler

начало

Алгоритм deleteRecord:

конец

получить MyPhoneRecord\* r

начало

освободить   
r->nickname, r

Алгоритм did\_user\_accept:

нет

нет

да

да

конец

Вернуть 0

Вернуть 1

!strcmp(answer, ”n”)

!strcmp(answer, ”y”)

Ввод answer

char answer[128]=” ”

Вывод “Are you sure? (y/n) ”

начало

# Выбор и обоснование типов переменных

В программе используется пользовательский тип данных – структура, синонимом имени типа которой выступает лексема MyPhoneRecord. Эта структура содержит другую пользовательскую структуру, синонимом имени типа которой является MyPhoneNumber number, и указатель на динамически создаваемую си-строку char\* nickname.

Структура MyPhoneNumber содержит 2 беззнаковых целых числа unsigned \_\_int64 number:50 и unsigned \_\_int64 plus:1. Первое поле содержит информацию о цифрах номера. Поскольку стандарт E.164 предусматривает телефонные номера длиной не более 15 цифр, то для представления такого номера будет достаточно 50 бит . Второе поле содержит информацию о том, есть ли в начале номера телефона “+”, поскольку значение этого поля можно считать предикатом, для него будет достаточно 1 бита.

Телефонная книга в программе хранится в виде массива указателей на записи MyPhoneRecord\* phonebook[100], для счетчиков используются переменные типа int, в качестве переменных для вводимых данных используются си-строки char[64], чего будет достаточно для имени контакта и телефонного номера.

# Вводимые и выводимые параметры и их типы

В программу, после приглашения для ввода подаются строки char[64], для подтверждения конца ввода – char[100].

Программа выводит:

* приглашение ко вводу имени char[18],
* приглашение ко вводу номера char[22],
* сообщение о конце ввода char[41],
* вопрос о конце ввода char[21],
* шапка таблицы char[51],
* номер строки таблицы в 3 символа int и пробел,
* две строки char[31] и char[21] – имя и телефонный номер контакта,
* системное сообщение (от system("pause")).

# Структура проекта

Файлы проекта содержатся в 2-х папках:

* libs:
  + 20.11.2021 12:24 3 453 myDecToBinLibFunctions.cpp
  + 18.11.2021 21:15 1 605 myDecToBinLibFunctions.h
  + 19.11.2021 19:07 2 682 myPhoneNumberLib.cpp
  + 18.11.2021 10:31 1 748 myPhoneNumberLib.h
* Lab4:
  + 20.11.2021 14:20 1 820 Lab4.cpp
  + 18.11.2021 00:29 7 458 Lab4.vcxproj
  + 18.11.2021 00:29 1 590 Lab4.vcxproj.filters
  + 17.11.2021 23:39 305 Lab4.vcxproj.user

# Текст программы

//Lab4.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS // MS VS do forbids unsafe functions like 'scanf'

#include<stdio.h> //for input-output

#include "../libs/myPhoneNumberLib.h" //for phone numbers

int main()

{

MyPhoneRecord\* phonebook[100] = { NULL }; //declarations and filling with '0'

int i = 0; //just a counter

do {

while (1) { // always

char name\_buffer[64], num\_buffer[64]; // buffers declaration

printf("Input a name\n>>> "); // prompt for name input

do {

gets\_s(name\_buffer, 64); // input string

} while (name\_buffer[0]=='\n' || name\_buffer[0]=='\0');

// checking if this is a null string

if (!strcmp(name\_buffer, "0")) { // if user inputed 0

break; // go to asking if user want to exit

}

printf("Input the number\n>>> "); // prompt for number input

do {

\_itoa(rand(),num\_buffer,10);

gets\_s(num\_buffer, 64); // input strings

} while (name\_buffer[0] == '\n' || name\_buffer[0] == '\0');

// checking if this is a null string

if (!strcmp(num\_buffer, "0")) { // if user inputed 0

break; // go to asking if user want to exit

} else {

phonebook[i++] = create\_record(name\_buffer, num\_buffer);

//else create record

}

}

printf("You inputed '0', you are going to exit.\n"); // ask about exiting

} while (!did\_user\_accept()); // checking iif user really want to exit

printf("# Nickname Number\n"); // table header

for (int j = 0; j < i; j++) { // for all records

printf("%4d", j+1); // print number

print\_record(phonebook[j], 30,' '); // print record

delete\_record(phonebook[j]); // delete (free) records memory

phonebook[j] = NULL; // erase pointer from array

}

do\_pause(); // do not close console

return 0;

}

//myPhoneNumberLib.h

#ifndef MY\_PHONE\_NUMBER\_LIB\_INCLUDED

#define MY\_PHONE\_NUMBER\_LIB\_INCLUDED

#define \_CRT\_SECURE\_NO\_WARNINGS // MS VS do forbids unsafe functions like 'scanf'

#include <stdio.h> // for input-output

#include <process.h> // for 'system' function

#include <string.h> // for string functions

#include <stdlib.h> // for 'atoi' functions

#include "myDecToBinLibFunctions.h"

// for functions from previous labs

typedef struct {

unsigned \_\_int64 number : 50; // is digits in a number

/\*

According to E.164 (standard phone number recording format)

phone numbers should be not longer than 15 digits.

So greatest phone number is: 999 999 999 999 999.

Then 50 bits are enough for recording this number,

because 2^50 = 1 125 899 906 842 624.

\*/

unsigned \_\_int64 plus : 1; // =1 if number includes +, else =0

}MyPhoneNumber;

typedef struct {

MyPhoneNumber number; // phone number

char\* nickname; // nickname

}MyPhoneRecord;

/// <summary>

/// Creates record, allocs memory for name

/// DO NOT FORGIVE DELETE PHONE RECORD

/// </summary>

/// <param name="nick">nickname</param>

/// <param name="number">phone number</param>

/// <returns>returns pointer to new record</returns>

MyPhoneRecord\* create\_record(const char\* nick, const char\* number);

/// <summary>

/// Deletes phone record by pointer

/// </summary>

/// <param name="r">pointer to record</param>

void delete\_record(MyPhoneRecord\* r);

/// <summary>

/// Print phone number record cutting name

/// </summary>

/// <param name="r">pointer to record</param>

/// <param name="nick\_len">name length</param>

void print\_record(const MyPhoneRecord\* r, int nick\_len, char filler=' ');

# #endif // !MY\_PHONE\_NUMBER\_LIB\_INCLUDED

//myPhoneNumberLib.h

#include "myPhoneNumberLib.h"

MyPhoneRecord\* create\_record(const char\* nick, const char\* number)

{

MyPhoneRecord\* res = new MyPhoneRecord; // allocate memory for record

res->nickname = new char[strlen(nick) + 1];//allocate memory for nickname and '\0'

strcpy(res->nickname, nick); // copy nick to record nickname

char\* filtered\_num = new char[strlen(number) + 1];

strcpy(filtered\_num, number);

// allocate memory for filtered number and '\0'

filter\_string(filtered\_num); // filter string

if (filtered\_num[0] == '+') { res->number.plus = 1; } // set plus

else { res->number.plus = 0; } // set no plus

if (filtered\_num[0] == '-') { filtered\_num[0] = '+'; }// ignore '-'

res->number.number = atoll(filtered\_num);

// extract int from c-string and put it to res

delete[] filtered\_num;

return res;

}

void delete\_record(MyPhoneRecord\* r)

{

delete r->nickname; // free c-string

delete r; // free record

}

void print\_record(const MyPhoneRecord\* r, int nick\_len, char filler)

{

char\* nick = new char[nick\_len + 1];// allocate memory for nickname in the table

strncpy(nick, r->nickname, nick\_len);// copy nickname to nick

nick[nick\_len] = '\0'; // end of string

inflate\_string(nick, nick\_len, StringAlign::left, filler); // formatting nickname

char number\_reverse[21]; // string for a number

unsigned \_\_int64 n = r->number.number; // temp variable for

//transforming number to reversed string

int tmp = 0; // just a counter

while (n) {

number\_reverse[tmp++] = n % 10 + '0';

// write right digit to the string and increase i

n /= 10; // move number to 1 digit right

}

if (r->number.plus) { // if number includes '+'

number\_reverse[tmp] = '+'; // add + to the string

number\_reverse[tmp + 1] = '\0'; // end of string

} else {

number\_reverse[tmp] = '\0'; // end of string

}

reverse\_string(number\_reverse); // reversing string

inflate\_string(number\_reverse, 20, StringAlign::right, filler);

//formatting number

printf("%s%s\n", nick, number\_reverse); // print row

delete[] nick; // free temp c-string

}

//myDecToBinLibFunctions.h

#ifndef myDecToBinLibFunctionsINCLUDED

#define myDecToBinLibFunctionsINCLUDED

#define MY\_STR\_SIZE 100 // size of the string, that user will be input

#define \_CRT\_SECURE\_NO\_WARNINGS // MS VS do forbids unsafe functions like 'scanf'

#include <stdbool.h> // standard bool type library

#include <string.h> // standard string functions library

#include <stdio.h> // standard input/output library

#include <process.h> // library for system functions

char\* filter\_string(char\* s);

/\*

removes all 'bad' characters (not digits or sign if it is first)

in the c-string

function is protected by limitation string size with MY\_STR\_SIZE

\*/

/// <summary>

/// reverses string

/// </summary>

/// <param name="s">pointer to c-string</param>

/// <returns></returns>

char\* reverse\_string(char\* s);

void represent\_as\_bin(int n, char\* b);

/\*

Represents -2^16 <= n < 2^16 in binary notation and writes

result to the c-string b, that MUST BE AT LEAST 32 SYBLOS LENGTH

\*/

bool did\_user\_accept();

/\*

Asks user if he really want to exit

returns true if it's true else returns false

\*/

void do\_pause(char print\_dir = 0);

/\*

Do not close console,

if print\_dir is not 0 - uses dir command

function is safe

\*/

enum StringAlign { left, center, right };

// types of the string align for inflate\_string function

char\* inflate\_string(

char\* s,

unsigned int width,

StringAlign align = StringAlign::left,

char filler = ' '

);

/\*

Increases string size with adding filler symbols according to align parameter

\*/

#endif

//myDecToBinLibFunctions.cpp

#include "myDecToBinLibFunctions.h"

char\* filter\_string(char\* s)

{

int i, // index of the char that we read from the string

j; // index of the char that we write to the string

for (i = j = 0; i < MY\_STR\_SIZE && s[i] != '\0'; i++) {

// for each char in the string

if (('0' <= s[i] && s[i] <= '9') || (j == 0 && (s[i] == '-'|| s[i]=='+'))){ // if char is not bad

s[j++] = s[i]; // we write it

}

}

s[j] = '\0'; // end of c-string

return s;

}

char\* reverse\_string(char\* s)

{

int n = strlen(s); // string size

for (int i = 0; i < n / 2; i++) { // for i from 0 to middle of the string

char tmp = s[i]; // just a buffer for values exchange

s[i] = s[n - i - 1]; // exchanging

s[n - i - 1] = tmp; // still exchanging

}

return s;

}

void represent\_as\_bin(int n, char\* b)

{

int i; // bit index

for (i = 0; i < 32; i++) { // i reaches from 0 to 32 increasing by 1

if ((n & (1 << (31 - i)))) { // if this bit is 1

// ^ use mask

// ^ set mask to 0...1...0, where 1 is at (31-i) bit

b[i] = '1'; // set current char in c-string to '1'

} else {

b[i] = '0'; // set current char in c-string to '0'

}

}

b[i] = '\0'; // end of c-string

}

void do\_pause(char print\_dir)

{

if (system(NULL)) {

if (print\_dir) {

system("DIR");

}

system("pause");

}

}

char\* inflate\_string(char\* s, unsigned int width, StringAlign align, char filler)

{

unsigned int n = strlen(s); // get length

switch (align) { // for the different aligns

case StringAlign::left:

{

int i;

for (i = n; i < width; i++) {

s[i] = filler; // just add fillers after end of

//string until they reach width

}

break;

}

case StringAlign::right:

{

int i;

for (i = width - 1; i >= 0; i--) {

s[i] = width - i <= n ? s[n - (width - i)] : filler;

// copy symbols to the end string if we have symbols left

// else just put fillers

}

break;

}

case StringAlign::center:

{

inflate\_string(s, (width - n) / 2 + n, StringAlign::right, filler);

// add fillers to the left

inflate\_string(s, width, StringAlign::left, filler); // add fillers to the right

break;

}

}

s[width] = '\0'; // end of string

return s;

}

bool did\_user\_accept()

{

char a[MY\_STR\_SIZE] = "a"; // c-string for answer

do {

printf("Are you sure (y/n): "); //prompt for input

scanf("%s", a); // get answer

// while answer is neither 'y' nor 'n'

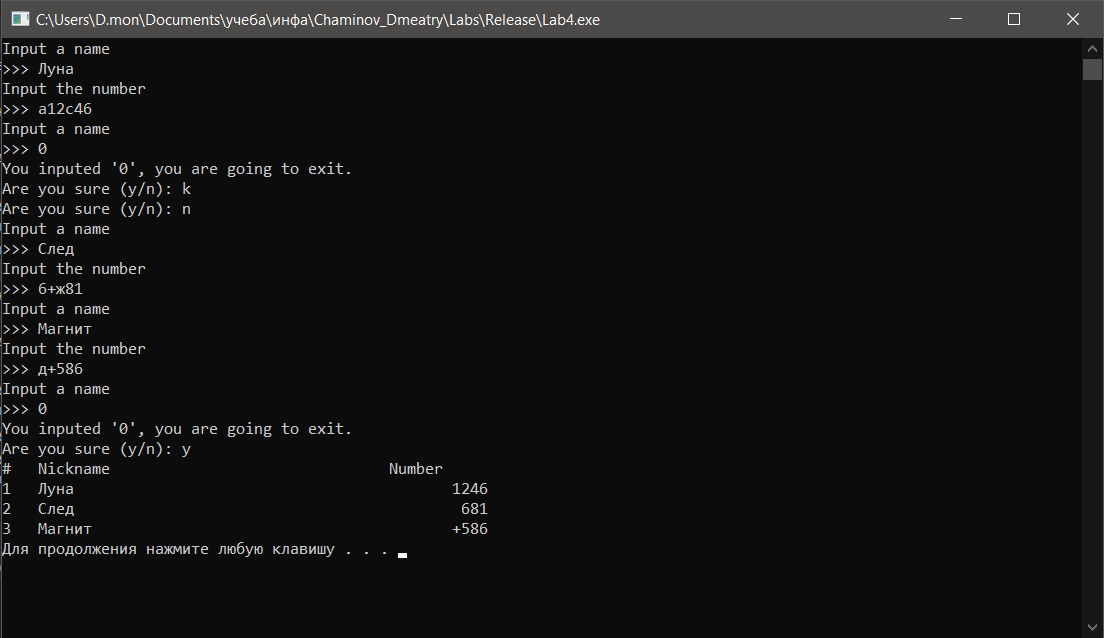
} while (!(!strncmp(a, "y", MY\_STR\_SIZE) || !strncmp(a, "n", MY\_STR\_SIZE)));

return !strcmp(a, "y"); // return if user want to exit

}

# Копии экрана при работе программы

Изображение выглядит как текст

Автоматически созданное описание 

# Контрольный пример, сравнение с эталоном

|  |  |
| --- | --- |
| Контрольные значения | |
| Входные | Выходные |
| Луна  а12с46  0  k  n  След  6+ж81  Магнит  д+586  0  y | |  |  |  | | --- | --- | --- | | 1 | Луна | 1246 | | 2 | След | 681 | | 3 | Магнит | +586 | |
| Реальные значения | |
| Входные | Выходные |
| Луна  а12с46  0  k  n  След  6+ж81  Магнит  д+586  0  y | # Nickname Number  1 Луна 1246  2 След 681  3 Магнит +586 |

Программа успешно прошла проверку.

# Результаты

Разработанная программа:

* Работает в операционных системах MS Windows 7 и выше.
* Является консольным приложением.
* Исполняется при запуске файла “Labs\Release\Lab4.exe”, который занимает 13кб памяти на диске.
* Принимает имена и телефонные номера контактов (не более 63 символов в строке входных данных) в количестве 100 и меньше и обрабатывает их.
* Работает корректно.
* Обладает следующими достоинствами:
  + Фильтрация ввода
  + Понятный интерфейс
  + Скорость работы
  + Имеет подтверждение завершения ввода
  + Удобный вывод в виде таблицы
* Обладает следующими недостатками:
  + Отсутствие защиты от переполнения (при вводе больших чисел в качестве номера возможно, а при вводе более 100 записей возможна ошибка сегментирования или непредсказуемое поведение программы).
  + Неупорядоченный вывод.
  + Интерфейс только на 1 языке.