**Курсовая работа**

**по дисциплине: «Основы программирования на языке C»**

2018 г

# Задание

Оглавление

[Задание 2](#_Toc516646898)

[Введение 4](#_Toc516646899)

[1 Исходные данные 5](#_Toc516646900)

[2 Глобальные данные 6](#_Toc516646901)

[3 Пользовательские макроопределения 7](#_Toc516646902)

[3.1 Описание подхода к тестированию программы 7](#_Toc516646903)

[3.2 Тестирование 7](#_Toc516646904)

[4 Пользовательские функции 8](#_Toc516646905)

[5 Внешние файлы 9](#_Toc516646906)

[6 Нестандартные функции 10](#_Toc516646907)

[7 Диагностические сообщения и действия оператора 11](#_Toc516646908)

[8 Описание порядка работы с программой 12](#_Toc516646909)

[Заключение 13](#_Toc516646910)

[Список литературы 14](#_Toc516646911)

[Приложения 15](#_Toc516646912)

# Введение

Данная работа выполнена в виде программы на языке программирования C в среде разработки C++ Builder.

Программа предназначена для работы с англо-русским словарём. Содержимое словаря хранится в файле dictionary.dat. Графический интерфейс выполнен в стиле MSDOS с использованием символов псевдографики. Он предоставляет полное управление словарём – хранение, отображение, добавление и изменение данных, поиск слов, сортировку по заданному полю, а также предоставление общей и контекстной справки.

# Исходные данные

# Глобальные данные

## Описание глобальных данных модуля main.c

Массив с описаниями элементов верхнего меню:

ITEM\_DEF top\_menu\_items[top\_item\_count];

Массив с описаниями элементов меню подтверждения выхода:

static ITEM\_DEF exit\_menu\_items[exit\_item\_count];

Массив заголовков основной таблицы:

char\* main\_headers[main\_column\_count];

Массив заголовков меню подтверждения выхода

char\* exit\_headers[exit\_column\_count];

код завершения для функций обратного вызова

int exit\_code = 0;

признак перезапуска основной таблицы

int redraw\_main = 1;

признак изменившихся данных

int data\_modified = 0;

признак необходимости сохранения данных

int save\_data = 0;

признак отмены выхода

int exit\_canceled = 0;

индекс для инициализации обновлённого меню

int initial\_table\_index = 0;

буфер для хранения настроек консоли:

CONSOLE\_SCREEN\_BUFFER\_INFO csbInfo;

экземпляр верхнего меню

MENU top\_menu;

указатель на экземпляр основной таблицы

MENU \*ptable = NULL;

## Описание глобальных данных модуля dict.c

Глобальная строковая константа – имя файла данных:

const char\* dict\_file\_name = "dictionary.dat";

глобальный экземпляр словаря (односвязный список):

LIST1 dict;

# Описание пользовательских макроопределений

## Макроопределения модуля main

Максимально допустимая длина строки:

#define MAX\_STRING 256

Пользовательские метки, чтобы различать меню в функциях обратного вызова:

* #define MENU\_TAG\_DEFAULT 0 – метка по умолчанию;
* #define MENU\_TAG\_TOP 1 – верхнее меню;
* #define MENU\_TAG\_MAIN 2 – основное меню (таблица данных) ;
* #define MENU\_TAG\_EXIT 3 – меню подтверждения выхода;
* #define MENU\_TAG\_HELP\_GLOBAL 4 – окно общей справки;
* #define MENU\_TAG\_SORT 6 – меню выбора поля для сортировки;

Количество пунктов верхнего меню

#define top\_item\_count 7

Количество пунктов верхнего меню

#define main\_column\_count 4

Количество ячеек (столбцов) основного табличного меню

#define exit\_column\_count 1

Количество пунктов меню подтверждения выхода

#define exit\_item\_count 3

## Макроопределения модуля dict

Максимальная длина хранимой в словаре строки:

#define MAX\_DICT\_STR\_LEN 32

Количество полей (Слово, Часть речи, Перевод):

#define DICT\_FLD\_CNT 3

## Макроопределения модуля menu

Признаки ориентации меню:

* #define MENU\_ORIENT\_HORZ 0 – горизонтальное
* #define MENU\_ORIENT\_VERT 1 – вертикальное

Максимальная длина строки заголовка

#define MAX\_MENU\_HDR 256 //

Флаги отрисовки меню и обработки сообщений:

* #define MENU\_FLAG\_WND 0x1 – отобразить окно
* #define MENU\_FLAG\_ITEMS 0x2 – отобразить элементы
* #define MENU\_DRAW\_SEL 0x4 – отобразить текущий элемент
* #define MENU\_NAVIGATOR 0x8 – обработка нажатия клавиш навигации и Enter
* #define MENU\_HOTKEYS 0x10 – обработка дополнительно заданных горячих клавиш
* #define MENU\_FULL\_FLAGS 0x1F – все флаги включены

Максимальное количество ячеек (столбцов вертикального меню)

#define MAX\_CELLS 50

Флаги прокрутки текущего пункта

* #define MENU\_CURR\_FWD 0 – на один пункт вперёд
* #define MENU\_CURR\_REV 1 – на один пункт назад

Символ фоновой зарисовки

#define MENU\_WHITESPACE ' '

Наибольшее из двух значений:

#define MAX(a, b) (((a) > (b)) ? (a) : (b))

## Макроопределения модуля input

Максимальная длина подписи поля

#define MAX\_TITLE 50

Предопределённые индексы для обращения к содержимому:

* #define TITLE 0 – заголовок
* #define BUFFER 1 – редактируемое поле данных

Общее количество колонок (заголовок и данные)

#define COLUMNS 2

## Глобальные макроопределения

В заголовочном файле cоdes.h объявлены макроопределения для кодов нажатых клавиш и символов псевдографики.

Коды клавиатуры:

* #define KEY\_ARROW\_UP 72 – стрелка вверх
* #define KEY\_ARROW\_LEFT 75 – стрелка влево
* #define KEY\_ARROW\_RIGHT 77 – стрелка вправо
* #define KEY\_ARROW\_DOWN 80 – стрелка вниз
* #define KEY\_TAB 9 – клавиша табуляции
* #define KEY\_ENTER 13 – забой (клавиша ввода)
* #define KEY\_ESC 27 – клавиша Esc
* #define KEY\_BACKSPACE 8 – клавиша Backspace
* #define KEY\_SPACE 32 – пробел
* #define KEY\_DEL 83 – клавиша Del
* #define KEY\_HOME 71 – клавиша Home
* #define KEY\_END 79 – клавиша End
* #define KEY\_F1 59 – клавиша F1
* #define KEY\_F2 60 – клавиша F2
* #define KEY\_F3 61 – клавиша F3
* #define KEY\_F4 62 – клавиша F4
* #define KEY\_F5 63 – клавиша F5
* #define KEY\_F6 64 – клавиша F6
* #define KEY\_F7 65 – клавиша F7
* #define KEY\_F8 66 – клавиша F8
* #define KEY\_F9 67 – клавиша F9
* #define KEY\_F10 68 – клавиша F10
* #define KEY\_F11 133 – клавиша F11
* #define KEY\_F12 134 – клавиша F12

Символы псевдографики:

* #define CHAR\_BORDER\_HORZ 205
* #define CHAR\_BORDER\_VERT 186
* #define CHAR\_BORDER\_LEFT\_TOP 201
* #define CHAR\_BORDER\_LEFT\_BOTTOM 200
* #define CHAR\_BORDER\_RIGHT\_TOP 187
* #define CHAR\_BORDER\_RIGHT\_BOTTOM 188
* #define CHAR\_BORDER\_LEFT\_JOINT 204
* #define CHAR\_BORDER\_RIGHT\_JOINT 185
* #define CHAR\_BORDER\_TOP\_JOINT 203
* #define CHAR\_BORDER\_BOTTOM\_JOINT 202
* #define CHAR\_BORDER\_CROSS\_JOINT 206

# Описание пользовательских функции

## Функции модуля main

int Run() – основная рабочая функция программы

int main(int argc, char\* argv[]) – точка входа программы.

**Функции обратного вызова.**

Обработчики пунктов верхнего меню:

* int Add(MENU\* pm, ITEM\*) – пункт «Добавить»
* int Edit(MENU\* menu, ITEM\*) – пункт «Изменить»
* int Delete(MENU\* menu) – пункт «Удалить»
* int Search(MENU\* menu, ITEM\*) – пункт «Поиск»
* int Sort(MENU\* menu, ITEM\*) – пункт «Сортировка»
* int Save(MENU\* menu, ITEM\*) – пункт «Сохранить»
* int Help(MENU\* menu, ITEM\*) – пункт «Справка»
* int Exit(MENU\* menu, ITEM\*) – пункт «Выйти»

Обработчики пунктов меню подтверждения выхода:

* int ExitYes(MENU\* menu, ITEM\* item) – выход с сохранением данных.
* int ExitNo(MENU\* menu, ITEM\* item) – выход без сохранения данных.
* int ExitCancel(MENU\* menu, ITEM\* item) – отмена выхода.

Обработчики горячих клавиш:

* int DefaultESC(MENU\* menu) – обработчик клавиши Esc по умолчанию, устанавливается для большинства меню и производит закрытие текущего меню без дополнительных действий.
* int ESC(MENU\* menu) – обработчик клавиши Esc для главного и верхнего меню, проверяет необходимость сохранения данных перед выходом.
* int F1(MENU\* menu) – обработчик горячей клавиши F1 (Справка).
* int F2(MENU\* menu) – обработчик горячей клавиши F2 (Сохранение).
* int F3(MENU\* menu) – обработчик горячей клавиши F3 (Поиск).
* int F9(MENU\* menu) – обработчик горячей клавиши F9 (Переход к верхнему меню).

Функции обратного вызова для односвязного списка:

* void curr\_menu\_changed(MENU\* menu, int direction, int wrap) – обработчик прокрутки меню на один элемент вперёд или назад.
* int dict\_entry\_display(void\* data, int index, void\* param) – перенос очередного элемента словаря в элемент описания пункта меню.
* int dict\_entry\_find(void\* data, void\* param) – проверка элемента словаря на соответствие критерию поиска.
* int dict\_entry\_compare(void\* data1, void\* data2, void\* arg) – сравнение двух элементов словаря.

**Вспомогательные функции.**

int HelpFromFile(HANDLE hStdOut, const char\* file\_name, const char\* title, SMALL\_RECT rect) – отображение текста справки из файла на всплывающем окне.

int ShowMenu(HANDLE hStdOut, ITEM\_DEF\* menu\_items, int item\_count, const char\* title, SMALL\_RECT rect, int flags, int user\_tag, ExecuteHotketCB f1CB) – отображение упрощённого вертикального меню в один столбец.

ITEM\_DEF\* MenuItemsFromFile(const char\* file\_name, int max\_count, int max\_len, int\* pcount) – загрузка пунктов меню из файла.

void LoadInitialData() – загрузка тестовых данных в словарь (отладочная функция).

## Функции модуля dict

DICT\_ENTRY\* dict\_entry\_new(const char\* word\_eng, const char\* word\_part, const char\* word\_rus) – создание нового элемента словаря.

int dict\_load(const char\* file\_name) – загрузка словаря из файла.

int dict\_entry\_save(void\* data, int index, void\* param) – сохранение одного элемента словаря в файл.

int dict\_save(const char\* file\_name) – сохранение словаря в файл.

void dict\_entry\_clear(DICT\_ENTRY\* entry) – удаление элемента словаря.

void dict\_clear() – удаление всего словаря из памяти.

## Функции модуля menu

int menu\_init(MENU\* menu, MENU\* parent, HANDLE hstdout,

ITEM\_DEF\* item\_defs, int item\_count, int cell\_count, int orient,

const SMALL\_RECT\* prect, int border, char\* headers[]) –

инициализация экземпляра меню.

int menu\_add\_hotkey(MENU\* menu, int code, ExecuteHotketCB cb) – добавить обработчик горячей клавиши.

void menu\_clear(MENU\* menu) – очистка экземпляра меню.

void menu\_fill\_wnd(MENU\* menu, int items\_count) – заполнение фона окна меню.

int menu\_draw(MENU\* menu, int flags) – основная оконная функция меню.

void menu\_active\_color(MENU\* menu, WORD attr) – настройка отображения текущего элемента меню.

void menu\_inactive\_color(MENU\* menu, WORD attr) настройка отображения неактивных элементов меню.

void menu\_prev(MENU\* menu) – выделить предыдущий пункт.

void menu\_next(MENU\* menu) – выделить следующий пункт.

void menu\_gotoxy(MENU\* menu, int x, int y) – переместить курсор по координатам x, y.

void itemMenu(MENU\* menu, int activate) – выделить пункт меню или снять выделение.

void saveCursorPosition(MENU\* menu) – сохранить текущую позицию курсора в переменной.

void menu\_cls(MENU\* menu) – отобразить фон окна меню.

void showCursor(MENU\* menu, int visible) – показать/убрать курсор.

void menu\_del\_curr(MENU\* menu) – удалить текущий элемент.

## Функции модуля input

int box\_init(InputBox\* box, HANDLE handle, SMALL\_RECT rect, char\*\*\* contents, int row\_count) – инициализация экземпляра окна ввода.

void box\_clear(InputBox\* box) – очистка полей экземпляра окна ввода.

int box\_save(InputBox\* box) – сохранить прямоугольник на экране.

int box\_draw(InputBox\* box) – нарисовать окно ввода и запустить цикл обработки нажатия клавиш.

int box\_restore(InputBox\* box) – восстановить прямоугольник на экране.

## Функции модуля list

LIST1\* list1\_new() – создать список.

int list1\_delete(LIST1\* list) – удалить список.

int list1\_init(LIST1\* list) – проинициализировать экземпляр списка

int list1\_clear(LIST1\* list) – очистить поля экземпляра списка.

int list1\_push\_front(LIST1\* list, void\* data) – добавить элемент в начало списка.

int list1\_push\_back(LIST1\* list, void\* item) – добавить элемент в конец списка.

int list1\_for\_each(LIST1\* list, LIST\_ITEM\_PROC find, void\* param) – перебор всех элементов с использованием функции обратного вызова.

void\* list1\_curr\_fwd(LIST1\* list, int wrap) – сместить текущий элемент на одну позицию вперёд.

void\* list1\_curr\_rev(LIST1\* list, int wrap) – сместить текущий элемент на одну позицию назад.

void\* list1\_front(LIST1\* list) – доступ к данным первого элемента.

void\* list1\_back(LIST1\* list) – доступ к данным последнего элемента.

void\* list1\_curr(LIST1\* list) – доступ к данным текущего элемента.

void\* list1\_erase(LIST1\* list, LIST\_ITEM\_FIND cb, void\* param) – удаление элементов, удовлетворяющих условиям функции обратного вызова.

void\* list1\_erase\_current(LIST1\* list) – удаление текущего элемента.

int list1\_get\_current\_index(LIST1\* list) – получение порядкового номера текущего элемента.

int list1\_set\_current\_index(LIST1\* list, int index) – установление текущего элемента по индексу.

int list1\_search(LIST1\* list, LIST\_ITEM\_FIND check, void\* param) – поиск элемента с использованием функции обратного вызова.

void list1\_sort(LIST1\* list, LIST\_ITEM\_COMP compare, void\* param) – сортировка списка с использованием функции обратного вызова.

# Нестандартные функции

# Внешние файлы

Внешние файлами являются файл данных, в котором хранятся англо-русский словарь, а также файлы общей и контекстной справки:

dictionary.dat

help.txt

sort.txt

# Диагностические сообщения и действия оператора

# Описание порядка работы с программой

# Заключение

В результате работы была спроектирована и разработана программная система, позволяющая работать с базой данный, представляющей собой англо-русский словарь.

В ходе работы над проектом получены практические навыки разработки программ на языке высокого уровня C. Изучено множество аспектов разработки программ на этом языке – в первую очередь, модульная организация сложной программной системы и управление памятью. Изучены особенности создания с помощью этого языка программирования продвинутого пользовательского интерфейса с использованием только консольного окна.

Кроме того, получены навыки работы в среде C++ Builder на чистом C и без использования библиотеки компонентов VCL.

# Список литературы

1. Подбельский В.В, Фомин С.С. Программирование на языке C / М.: «Финансы и статистика». 2003.

2. Дейтел Х.М., Дейтел П.Дж. Как программировать на C / М: «БИНОМ», 2000.

3. Шилдт, Г. Справочник по C. Диасофт М.: 2000.

4. ГОСТ 2.105-95 Общие требования к текстовым документам.

# Приложения

main.h