**Министерство образования и науки Российской федерации**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» им. В. И. Ульянова (Ленина)» (СПбГЭТУ «ЛЭТИ»)**

**Факультет компьютерных технологий и информатики**

**Кафедра Вычислительной техники**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой**

**по дисциплине «Программирование. Дополнительные главы»**

Тема: **«**Электронная картотека**»**

Выполнил студент гр.5307 Самоуков Н.В.

Проверил: к.т.н., доцент кафедры ВТ Сискович Т.И.

Санкт-Петербург

2016

Оглавление

[Цель 4](#_Toc451728864)

[1. Задание 4](#_Toc451728865)

[2. Уточнение задания 4](#_Toc451728866)

[3. Контрольные примеры 4](#_Toc451728867)

[4. Описание структур 6](#_Toc451728868)

[4.1. Описание полей структуры DATA 6](#_Toc451728869)

[4.2. Описание полей структуры OSHIP 6](#_Toc451728870)

[4.3. Описание полей структуры SORTDATA 6](#_Toc451728871)

[4.4. Описание полей структуры FINDDATA 7](#_Toc451728872)

[5. Краткое описание алгоритма 8](#_Toc451728873)

[6. Иерархическая структура программы 9](#_Toc451728874)

[7. Описание функций 9](#_Toc451728875)

[7.1. Функция main 9](#_Toc451728876)

[7.2. Функция input\_number 9](#_Toc451728877)

[7.3. Функция input\_yn 10](#_Toc451728878)

[7.4. Функция is\_allowed 10](#_Toc451728879)

[7.5. Функция input\_string 11](#_Toc451728880)

[7.6. Функция str\_len 12](#_Toc451728881)

[7.7. Функция is\_sm\_letter 12](#_Toc451728882)

[7.8. Функция is\_bi\_letter 13](#_Toc451728883)

[7.9. Функция correct\_name 13](#_Toc451728884)

[7.10. Функция get\_nt 14](#_Toc451728885)

[7.11. Функция get\_path 15](#_Toc451728886)

[7.12. Функция get\_type 15](#_Toc451728887)

[7.13. Функция get\_coord 16](#_Toc451728888)

[7.14. Функция get\_sized 17](#_Toc451728889)

[7.15. Функция get\_struct\_info 18](#_Toc451728890)

[7.16. Функция O\_get\_structs 18](#_Toc451728891)

[7.17. Функция O\_print\_struct 19](#_Toc451728892)

[7.18. Функция O\_print 19](#_Toc451728893)

[7.19. Функция O\_free\_all 20](#_Toc451728894)

[7.20. Функция O\_get\_len 21](#_Toc451728895)

[7.21. Функция F\_gets 21](#_Toc451728896)

[7.22. Функция O\_add\_after 22](#_Toc451728897)

[7.23. Функция test\_file 23](#_Toc451728898)

[7.24. Функция save\_good\_placed 23](#_Toc451728899)

[7.25. Функция save 24](#_Toc451728900)

[7.26. Функция load 25](#_Toc451728901)

[7.27. Функция select\_sort 25](#_Toc451728902)

[7.28. Функция select\_find 26](#_Toc451728903)

[7.29. Функция a\_sma\_b 27](#_Toc451728904)

[7.30. Функция is\_need 27](#_Toc451728905)

[7.31. Функция O\_add\_frist 28](#_Toc451728906)

[7.32. Функция O\_insert\_spis 29](#_Toc451728907)

[7.33. Функция input\_menu 29](#_Toc451728908)

[7.34. Функция output\_menu 30](#_Toc451728909)

[7.35. Функция remove\_menu 31](#_Toc451728910)

[7.36. Функция help\_menu 31](#_Toc451728911)

[7.37. Функция O\_create\_copy 32](#_Toc451728912)

[7.38. Функция O\_find 33](#_Toc451728913)

[7.39. Функция O\_insert 33](#_Toc451728914)

[7.40. Функция O\_sort 34](#_Toc451728915)

[7.41. Функция calculate\_menu 35](#_Toc451728916)

[7.42. Функция edit\_one 35](#_Toc451728917)

[7.43. Функция charg\_menu 36](#_Toc451728918)

[7.44. Функция O\_del\_one 37](#_Toc451728919)

[7.45. Функция del\_menu 37](#_Toc451728920)

[7.46. Функция edit\_menu 38](#_Toc451728921)

[7.47. Функция my\_getch 39](#_Toc451728922)

[8. Исходный код программы 39](#_Toc451728923)

[9. Результат 75](#_Toc451728924)

[Вывод 75](#_Toc451728925)

**Цель**

Получить практические навыки по созданию электронных картотек на языке С\С++.

**1. Задание**

Создать электронную картотеку, хранящуюся на диске, и программу, обеспечивающую взаимодействие с ней.

**2. Уточнение задания**

Предметная область кораблики. Данные о кораблике это: имя, тип, положение по x, положение по y, положение по z, водоизмещение, скорость. Программа должна загружать и сохранять в файл информационные поля элементов односвязного списка. Также предусмотреть возможность изменения данных одного элемента и ввод данных с клавиатуры.

**3. Контрольные примеры**

Контрольные примеры представлены в таблице 1.

*Таблица 1. Контрольные примеры*

|  |  |  |
| --- | --- | --- |
| **№ Примера** | **Входные данные** | **Ожидаемый результат** |
| 1 | В файле t.txt:  *Поиск условия общего равновесия в макроэкономики основывается на законе:*  *Л. Вальраса*  Чтение из файла | Сообщение: ”Не удалось загрузить список” |
| 2 | 1)ship={"Ship","Unknown",7,7,7  ,60,100}  Запись в файл t.txt | В файле t.txt:  *Ship*  *Unknown*  *7*  *7*  *7*  *60*  *100* |
| 3 | В файле t.txt:  *Ship I*  *Unknown*  *3*  *3*  *3*  *400*  *500*  *Ship X*  *Pinnace*  *4*  *7*  *-10*  *500*  *600*  Чтение из файла | 1)ship ={"Ship I"," Unknown",3,3,3,400,500}  2)ship ={"Ship X"," Pinnace",4,7,-10,500 ,600} |

**4. Описание структур**

**4.1. Описание полей структуры DATA**

Описание полей структуры DATA, приведено в таблице 2.

*Таблица 2. Описание полей структуры DATA.*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение** |
| name | char \* | Указатель на первый символ строки, содержащий имя кораблика |
| type | char \* | Указатель на первый символ строки, содержащий тип кораблика |
| x | int | Положение кораблика по x |
| y | int | Положение кораблика по y |
| z | int | Положение кораблика по z |

**4.2. Описание полей структуры OSHIP**

Описание полей структуры OSHIP, приведено в таблице 3.

*Таблица 3. Описание полей структуры OSHIP.*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение** |
| info | DATA | Данные о кораблике |
| link1 | OSHIP \* | Указатель на следующий элемент |

# 4.3. Описание полей структуры SORTDATA

Описание полей структуры SORTDATA ,приведено в таблице 4 Описание полей структуры SORTDATA.

Таблица 4. Описание полей структуры SORTDATA.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| s\_name | char | Тип сортировки по имени(1-по возрастанию 2-по убыванию) |
| s\_type | char | Тип сортировки по типу(1-по возрастанию 2-по убыванию) |
| s\_x | char | Тип сортировки по x(1-по возрастанию 2-по убыванию) |
| s\_y | char | Тип сортировки по y(1-по возрастанию 2-по убыванию) |
| s\_z | char | Тип сортировки по z(1-по возрастанию 2-по убыванию) |
| typ | char | Массив размера 5, в котором элемент означает критерий сортировки (1-имя 2-тип 3-x 4-y 5-z 0-нет критерия) |

# 4.4. Описание полей структуры FINDDATA

Описание полей структуры FINDDATA ,приведено в таблице 5 Описание полей структуры FINDDATA.

Таблица 5. Описание полей структуры FINDDATA.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| name | char \* | Указатель на первый символ строки, содержащий имя кораблика |
| type | char \* | Указатель на первый символ строки, содержащий тип кораблика |
| x | int | Положение кораблика по x |
| y | int | Положение кораблика по y |
| z | int | Положение кораблика по z |
| s\_name | char | Тип поиска по имени(0-не важно 1-те, что больше 2-те, что меньше 3-полное совпадение) |
| s\_type | char | Тип поиска по типу(0-не важно 1-те, что больше 2-те, что меньше 3-полное совпадение) |
| s\_x | char | Тип поиска по x(0-не важно 1-те, что больше 2-те, что меньше 3-полное совпадение) |
| s\_y | char | Тип поиска по y(0-не важно 1-те, что больше 2-те, что меньше 3-полное совпадение) |
| s\_z | char | Тип поиска по z(0-не важно 1-те, что больше 2-те, что меньше 3-полное совпадение) |

**5. Краткое описание алгоритма**

Шаг 1. Обнуяем указатель на первый элемент списка.

Шаг 2. Очищаем экран.

Шаг 3. Выводим меню.

Шаг 4. Вводим menu

Шаг 5. Очищаем экран

Шаг 7. Если menu == 1, то вызываем функцию создания меню ввода данных и Шаг 14.

Шаг 7. Если menu == 2, то вызываем функцию создания меню вывода данных и Шаг 14.

Шаг 8. Если menu == 3, то вызываем функцию создания меню обработки и Шаг 14.

Шаг 9. Если menu == 3, то вызываем функцию создания меню редактирования данных и

Шаг 14.

Шаг 10. Если menu == 3, то вызываем функцию создания меню удаления и Шаг 14.

Шаг 11. Если menu == 4, то вызываем функцию создания меню справки и Шаг 14.

Шаг 12. Если menu == 0, то Шаг 14.

Шаг 13. Вывести сообщение об ошибке.

Шаг 14.Если menu != 0,то Шаг 2, иначе конец программы.

**6. Иерархическая структура программы**

Иерархическая структура программы представлена на рисунке 1.

Рисунок 1. Иерархическая структура программы

**7. Описание функций**

**7.1. Функция main**

**Назначение функции**  
Функция для создания главного меню.  
  
**Описание переменных**  
Описание переменных представлено в таблице 6.

*Таблица 6. Переменные функции main*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Локальные параметры** | menu | int | Выбранный пункт меню |
| ship | OSHIP \* | Указатель на первый элемент списка |

**7.2. Функция input\_number**

**Назначение функции**  
Функция для ввода целого числа.  
  
**Прототип**  
int input\_number();  
  
**Возвращаемое значение**  
Введённое число.  
  
**Пример вызова**  
menu = input\_number();  
Фактические параметры указаны в пункте 7.46  
  
**Описание переменных**  
Описание переменных представлено в таблице 7.

*Таблица 7. Переменные функции input\_number*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Локальные параметры** | number | int | Введённое число |
| s | char \* | Указатель на первый символ исходной строки |
| length | int | Длина строки |
| ch | char | Введённый символ |

**7.3. Функция input\_yn**

**Назначение функции**  
Функция для ввода ответа y/n.  
  
**Прототип**  
bool input\_yn();  
  
**Возвращаемое значение**  
1 - 'y', 0 - 'n'.  
  
**Пример вызова**  
if (input\_yn())  
  
**Описание переменных**  
Описание переменных представлено в таблице 8.

*Таблица 8. Переменные функции input\_yn*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Локальные параметры** | sy | char | Введённый символ |
| ch | char | Нажатая клавиша |

**7.4. Функция is\_allowed**

**Назначение функции**  
Функция для проверки, является ли разрешённым символ.  
  
**Прототип**  
bool is\_allowed(char ch, char \*not\_allowed);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
1 - является, 0 - не является.  
  
**Пример вызова**  
if (is\_allowed(ch, not\_allowed) && length <= size && (ch != ' ' || length > 1 && s[length - 2] != ' '))  
Фактические параметры указаны в пункте 7.5  
  
**Описание переменных**  
Описание переменных представлено в таблице 9.

*Таблица 9. Переменные функции is\_allowed*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ch | char | Введённый символ |
| not\_allowed | char \* | Указатель на первый элемент массива данных с запрещёнными символами |
| **Локальные параметры** | b | bool | Флажок совпадения символа с запрещённым |

**7.5. Функция input\_string**

**Назначение функции**  
Функция для ввода строки.  
  
**Прототип**  
char \*input\_string(int size,char \*not\_allowed);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Указатель на первый символ введённой строки.  
  
**Пример вызова**  
file\_name = input\_string(70, "");  
Фактические параметры указаны в пункте 7.11  
  
**Описание переменных**  
Описание переменных представлено в таблице 10.

*Таблица 10. Переменные функции input\_string*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | size | int | Максимальный размер вводимой строки |
| not\_allowed | char \* | Указатель на первый символ строки с запрещёнными символами |
| **Локальные параметры** | s | char \* | Указатель на первый символ исходной строки |
| length | int | Длина строки |
| ch | char | Введённый символ |

**7.6. Функция str\_len**

**Назначение функции**  
Функция для вычисления длины строки.  
  
**Прототип**  
int str\_len(char\*s);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Длина строки.  
  
**Пример вызова**  
len = str\_len(ship->info.type);  
Фактические параметры указаны в пункте 7.37  
  
**Описание переменных**  
Описание переменных представлено в таблице 11.

*Таблица 11. Переменные функции str\_len*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | s | char\* | Указатель на первый элемент исходной строки |
| **Локальные параметры** | i | int | Длина строки |

**7.7. Функция is\_sm\_letter**

**Назначение функции**  
Функция для проверки, является ли символ прописной буквой.  
  
**Прототип**  
bool is\_sm\_letter(char ch);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
1 - является, 0 - не является.  
  
**Пример вызова**  
is\_p\_let = is\_sm\_letter(\*s) || is\_bi\_letter(\*s);  
Фактические параметры указаны в пункте 7.9  
  
**Описание переменных**  
Описание переменных представлено в таблице 12.

*Таблица 12. Переменные функции is\_sm\_letter*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ch | char | Проверяемый символ |

**7.8. Функция is\_bi\_letter**

**Назначение функции**  
Функция для проверки, является ли символ строчной буквой.  
  
**Прототип**  
bool is\_bi\_letter(char ch);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
1 - является, 0 - не является.  
  
**Пример вызова**  
is\_p\_let = is\_sm\_letter(\*s) || is\_bi\_letter(\*s);  
Фактические параметры указаны в пункте 7.9  
  
**Описание переменных**  
Описание переменных представлено в таблице 13.

*Таблица 13. Переменные функции is\_bi\_letter*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ch | char | Проверяемый символ |

**7.9. Функция correct\_name**

**Назначение функции**  
Функция для коррекции введённой строки.  
  
**Прототип**  
void correct\_name(char\* s);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
correct\_name(s);  
Фактические параметры указаны в пункте 7.10  
  
**Описание переменных**  
Описание переменных представлено в таблице 14.

*Таблица 14. Переменные функции correct\_name*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | s | char\* | Указатель на первый символ имени |
| **Локальные параметры** | ra | char | Разница в кодах прописных и строчных букв |
| is\_p\_let | bool | Флажок предыдущего символа(1-предыдущий символ буква 0-предыдущий символ не буква) |

**7.10. Функция get\_nt**

**Назначение функции**  
Функция для ввода имени.  
  
**Прототип**  
char\* get\_nt(char \*title,char\* name);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Указатель на первый символ введённой строки.  
  
**Пример вызова**  
ship->info.name = get\_nt("Ввод данных для сравнения", "имя");  
Фактические параметры указаны в пункте 7.42  
  
**Описание переменных**  
Описание переменных представлено в таблице 15.

*Таблица 15. Переменные функции get\_nt*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | title | char \* | Указатель на первый элемент строки с заголовком для ввода |
| name | char\* | Указатель на первый элемент строки с именем того, что вводим |
| **Локальные параметры** | s | char \* | Указатель на первый символ имени |
| exit | bool | Флажок выхода |

**7.11. Функция get\_path**

**Назначение функции**  
Функция для ввода пути к файлу.  
  
**Прототип**  
char\* get\_path(char\* demo\_f);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Указатель на первый символ введённой строки.  
  
**Пример вызова**  
file\_name = get\_path("out.txt");  
Фактические параметры указаны в пункте 7.35  
  
**Описание переменных**  
Описание переменных представлено в таблице 16.

*Таблица 16. Переменные функции get\_path*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | demo\_f | char\* | Указатель на первый элемент строки с путём к демо файлу |
| **Локальные параметры** | file\_name | char\* | Указатель на первый символ строки, содержащей путь к файлу |

**7.12. Функция get\_type**

**Назначение функции**  
Функция для ввода типа кораблика.  
  
**Прототип**  
char\* get\_type(char\* title);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Указатель на первый символ введённой строки.  
  
**Пример вызова**  
ship->info.type = get\_type("Ввод данных для сравнения");  
Фактические параметры указаны в пункте 7.42  
  
**Описание переменных**  
Описание переменных представлено в таблице 17.

*Таблица 17. Переменные функции get\_type*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | title | char\* | Указатель на первый элемент строки с заголовком для ввода |
| **Локальные параметры** | i | int | Вспомогательная переменная |
| s | char \* | Указатель на первый символ вводимой строки |
| type | char | Тип кораблика |
| len | char | Длина строки |

**7.13. Функция get\_coord**

**Назначение функции**  
Функция для ввода координат.  
  
**Прототип**  
int get\_coord(char\* title,char ch);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Введённая координата.  
  
**Пример вызова**  
ship->info.z = get\_coord("Ввод данных для сравнения", 'z');  
Фактические параметры указаны в пункте 7.42  
  
**Описание переменных**  
Описание переменных представлено в таблице 18.

*Таблица 18. Переменные функции get\_coord*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | title | char\* | Указатель на первый элемент строки с заголовком для ввода |
| ch | char | Символ координат, например 'x' |

**7.14. Функция get\_sized**

**Назначение функции**  
Это функция для ввода числа в заданных пределах.  
  
**Прототип**  
int get\_sized(char\* title,int min, int max, char\* name);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает введённое число.  
  
**Пример вызова**  
m = get\_sized("Удаление элементов картотеки с n до m", n, len, "m");  
Фактические параметры указаны в пункте 7.45  
  
**Описание переменных**  
Описание переменных представлено в таблице 19.

*Таблица 19. Переменные функции get\_sized*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | title | char\* | Указатель на первый элемент строки с заголовком для ввода |
| min | int | Минимальное значение для вводимого числа |
| max | int | Минимальное значение для вводимого числа |
| name | char\* | Указатель на первый элемент строки с именем того, что вводим |
| **Локальные параметры** | num | int | Введённое число |

**7.15. Функция get\_struct\_info**

**Назначение функции**  
Функция получения полей структуры.  
  
**Прототип**  
void get\_struct\_info(DATA \*info);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
get\_struct\_info(&(n\_el->info));  
Фактические параметры указаны в пункте 7.16  
  
**Описание переменных**  
Описание переменных представлено в таблице 20.

*Таблица 20. Переменные функции get\_struct\_info*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | info | DATA \* | Указатель на данные о кораблике |

**7.16. Функция O\_get\_structs**

**Назначение функции**  
Функция для ввода списка.  
  
**Прототип**  
OSHIP \*O\_get\_structs();  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
nsp = O\_get\_structs();  
Фактические параметры указаны в пункте 7.33  
  
**Описание переменных**  
Описание переменных представлено в таблице 21.

*Таблица 21. Переменные функции O\_get\_structs*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Локальные параметры** | n\_sp | OSHIP\* | Указатель на первый элемент нового списка |
| n\_el | OSHIP \* | Указатель на новый элемент |
| last\_el | OSHIP \* | Указатель на последний элемент списка |
| is\_end\_input | bool | Флажок выхода |

**7.17. Функция O\_print\_struct**

**Назначение функции**  
Функция для вывода информационных полей одной структуры.  
  
**Прототип**  
void O\_print\_struct(OSHIP \*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
O\_print\_struct(ship);  
Фактические параметры указаны в пункте 7.42  
  
**Описание переменных**  
Описание переменных представлено в таблице 22.

*Таблица 22. Переменные функции O\_print\_struct*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на редактируемый элемент |

**7.18. Функция O\_print**

**Назначение функции**  
Функция для вывода списка.  
  
**Прототип**  
void O\_print(OSHIP \*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
O\_print(ship);  
Фактические параметры указаны в пункте 7.34  
  
**Описание переменных**  
Описание переменных представлено в таблице 23.

*Таблица 23. Переменные функции O\_print*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент списка с результатами поиска |
| **Локальные параметры** | i | int | Номер кораблика |

**7.19. Функция O\_free\_all**

**Назначение функции**  
Функция для освобождения памяти из-под всего списка.  
  
**Прототип**  
OSHIP\* O\_free\_all(OSHIP\* ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает NULL.  
  
**Пример вызова**  
ship\_res = O\_free\_all(ship\_res);  
Фактические параметры указаны в пункте 7.41  
  
**Описание переменных**  
Описание переменных представлено в таблице 24.

*Таблица 24. Переменные функции O\_free\_all*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка с результатами поиска |
| **Локальные параметры** | link | OSHIP \* | Указатель на элемент, из-под которого освобождается память |

**7.20. Функция O\_get\_len**

**Назначение функции**  
Это функция для подсчёта количества элементов в списке.  
  
**Прототип**  
int O\_get\_len(OSHIP\* ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает количество элементов в списке.  
  
**Пример вызова**  
len = O\_get\_len(ship);  
Фактические параметры указаны в пункте 7.45  
  
**Описание переменных**  
Описание переменных представлено в таблице 25.

*Таблица 25. Переменные функции O\_get\_len*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные параметры** | i | int | Количество элементов в списке |

**7.21. Функция F\_gets**

**Назначение функции**  
Функция для считывания из файла строки.  
  
**Прототип**  
char\* F\_gets(char end, int maxlen, FILE\* f);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый символ считанной строки.  
  
**Пример вызова**  
if ((s = F\_gets('\n', 50, f)) != NULL)  
Фактические параметры указаны в пункте 7.26  
  
**Описание переменных**  
Описание переменных представлено в таблице 26.

*Таблица 26. Переменные функции F\_gets*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | end | char | Символ признака конца строки |
| maxlen | int | Максимальная Длина строки |
| f | FILE\* | Указатель на данные о файле |
| **Локальные параметры** | s | char \* | Указатель на первый символ считываемой строки |
| exit | bool | Флажок выхода |
| len | int | Длина строки |

**7.22. Функция O\_add\_after**

**Назначение функции**  
Это функция для добавления списка после определённого элемента исходного списка.  
  
**Прототип**  
void O\_add\_after(OSHIP \*sp, OSHIP \*add\_sp);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
O\_add\_after(ship, el);  
Фактические параметры указаны в пункте 7.39  
  
**Описание переменных**  
Описание переменных представлено в таблице 27.

*Таблица 27. Переменные функции O\_add\_after*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | sp | OSHIP \* | Указатель на первый элемент отсортированного списка |
| add\_sp | OSHIP \* | Указатель на вставляемый элемент |
| **Локальные параметры** | li | OSHIP \* | Вспомогательный указатель |

**7.23. Функция test\_file**

**Назначение функции**  
Функция для проверки правильности данных в файле.  
  
**Прототип**  
int test\_file(char\*file\_name);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает 1 - если данные правильны, 0 - если файл пуст, или его не существует, -1 - если данные в файле повреждены.  
  
**Пример вызова**  
if (test\_file(file\_name) != -1)  
Фактические параметры указаны в пункте 7.34  
  
**Описание переменных**  
Описание переменных представлено в таблице 28.

*Таблица 28. Переменные функции test\_file*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | file\_name | char\* | Указатель на первый символ строки пути к файлу |
| **Локальные параметры** | f | FILE\* | Указатель на данные о файле |
| s | char\* | Указатель на первый символ строки |
| i | int | Вспомогательная переменная |

**7.24. Функция save\_good\_placed**

**Назначение функции**  
Функция для вывода информационных полей в файл в "красивом" виде.  
  
**Прототип**  
bool save\_good\_placed(char\*file\_name, OSHIP\*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает 1 - если удалось записать, иначе 0.  
  
**Пример вызова**  
save\_good\_placed(file\_name, ship);  
Фактические параметры указаны в пункте 7.34  
  
**Описание переменных**  
Описание переменных представлено в таблице 29.

*Таблица 29. Переменные функции save\_good\_placed*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | file\_name | char\* | Указатель на первый символ строки пути к файлу |
| ship | OSHIP\* | Указатель на первый элемент списка с результатами поиска |
| **Локальные параметры** | f | FILE\* | Указатель на данные о файле |

**7.25. Функция save**

**Назначение функции**  
Функция для вывода информационных полей в файл.  
  
**Прототип**  
bool save(char\*file\_name, OSHIP\*ship, bool is\_add);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает 1 - если удалось записать, иначе 0.  
  
**Пример вызова**  
save(file\_name, ship, 1);  
Фактические параметры указаны в пункте 7.34  
  
**Описание переменных**  
Описание переменных представлено в таблице 30.

*Таблица 30. Переменные функции save*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | file\_name | char\* | Указатель на первый символ строки пути к файлу |
| ship | OSHIP\* | Указатель на первый элемент списка с результатами поиска |
| is\_add | bool | Флажок записи в режиме добавления |
| **Локальные параметры** | f | FILE\* | Указатель на данные о файле |
| link | OSHIP\* | Указатель на сохраняемый элемент |

**7.26. Функция load**

**Назначение функции**  
Функция для ввода данных из файла.  
  
**Прототип**  
OSHIP \*load(char \*file\_name);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент загруженного списка.  
  
**Пример вызова**  
nsp = load(file\_name);  
Фактические параметры указаны в пункте 7.33  
  
**Описание переменных**  
Описание переменных представлено в таблице 31.

*Таблица 31. Переменные функции load*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | file\_name | char \* | Указатель на первый символ строки пути к файлу |
| **Локальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| el | OSHIP\* | Вспомогательный указатель |
| last | OSHIP\* | Указатель на последний элемент списка |
| f | FILE \* | Указатель на данные о файле |
| err | bool | Флажок ошибки |
| end | bool | Флажок конца файла |
| s | char \* | Указатель на первый символ считанный элемент |

**7.27. Функция select\_sort**

**Назначение функции**  
Это функция для выбора типа сортировки.  
  
**Прототип**  
void select\_sort(SORTDATA \*sort);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
select\_sort(&sort);  
Фактические параметры указаны в пункте 7.41  
  
**Описание переменных**  
Описание переменных представлено в таблице 32.

*Таблица 32. Переменные функции select\_sort*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | sort | SORTDATA \* | Указатель на структуру с параметрами сортировки |
| **Локальные параметры** | b | bool | Вспомогательный флажок |
| i | int | Вспомогательная переменная |
| len | int | Количество критериев |
| r | int | Вспомогательная переменная |
| o | int | Вспомогательная переменная |

**7.28. Функция select\_find**

**Назначение функции**  
Это функция для выбора типа поиска.  
  
**Прототип**  
void select\_find(FINDDATA \*find);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
select\_find(&find);  
Фактические параметры указаны в пункте 7.41  
  
**Описание переменных**  
Описание переменных представлено в таблице 33.

*Таблица 33. Переменные функции select\_find*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | find | FINDDATA \* | Указатель на структуру с параметрами поиска |
| **Локальные параметры** | i | int | Вспомогательная переменная |
| len | int | Количество полей, по которым будет проводиться поиск |
| r | int | Вспомогательная переменная |
| typ[7] | char | Массив размера 7 данных о сортировке |
| o | int | Вспомогательная переменная |

**7.29. Функция a\_sma\_b**

**Назначение функции**  
Это функция для сравнения двух элементов.  
  
**Прототип**  
bool a\_sma\_b(DATA a, DATA b, SORTDATA \* sort\_t);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает 1 если a меньше b, иначе 0.  
  
**Пример вызова**  
if (a\_sma\_b(ship->link1->info, el->info, sort))  
Фактические параметры указаны в пункте 7.39  
  
**Описание переменных**  
Описание переменных представлено в таблице 34.

*Таблица 34. Переменные функции a\_sma\_b*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | a | DATA | Данные элемента a |
| b | DATA | Данные элемента b |
| sort\_t | SORTDATA \* | Указатель на структуру с параметрами сортировки |
| **Локальные параметры** | state | char | Значение результата(1 - a < b, 0 - a == b, -1 - a > b) |
| i | int | Вспомогательная переменная |

**7.30. Функция is\_need**

**Назначение функции**  
Функция для определения, подходит ли элемент критериям поиска.  
  
**Прототип**  
bool is\_need(DATA info, FINDDATA \*find);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает 1 если подходит, иначе 0.  
  
**Пример вызова**  
if (is\_need(ship->info, find))  
Фактические параметры указаны в пункте 7.38  
  
**Описание переменных**  
Описание переменных представлено в таблице 35.

*Таблица 35. Переменные функции is\_need*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | info | DATA | Данные элемента |
| find | FINDDATA \* | Указатель на структуру с параметрами поиска |

**7.31. Функция O\_add\_frist**

**Назначение функции**  
Это функция для добавления списка в начало исходного списка.  
  
**Прототип**  
OSHIP\* O\_add\_frist(OSHIP \*sp, OSHIP \*add\_sp);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship = O\_add\_frist(ship, nsp);  
Фактические параметры указаны в пункте 7.32  
  
**Описание переменных**  
Описание переменных представлено в таблице 36.

*Таблица 36. Переменные функции O\_add\_frist*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | sp | OSHIP \* | Указатель на первый элемент списка |
| add\_sp | OSHIP \* | Указатель на первый элемент введённого списка |
| **Локальные параметры** | li | OSHIP \* | Вспомогательный указатель |

**7.32. Функция O\_insert\_spis**

**Назначение функции**  
Это функция для добавления списка в указаное место.  
  
**Прототип**  
OSHIP \*O\_insert\_spis(OSHIP \*ship, OSHIP \*nsp);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship = O\_insert\_spis(ship, nsp);  
Фактические параметры указаны в пункте 7.33  
  
**Описание переменных**  
Описание переменных представлено в таблице 37.

*Таблица 37. Переменные функции O\_insert\_spis*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент списка |
| nsp | OSHIP \* | Указатель на первый элемент введённого списка |
| **Локальные параметры** | menu | int | Выбранный пункт меню |
| n | int | Номер элемента |
| len | int | Количество элементов в списке |
| link | OSHIP \* | Указатель на вставляемый элемент |
| file\_name | char\* | Указатель на первый символ строки, содержащей путь к файлу |

**7.33. Функция input\_menu**

**Назначение функции**  
Функция для создания меню ввода.  
  
**Прототип**  
OSHIP \*input\_menu(OSHIP \*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент введённого списка.  
  
**Пример вызова**  
ship = input\_menu(ship);  
Фактические параметры указаны в пункте 7.1  
  
**Описание переменных**  
Описание переменных представлено в таблице 38.

*Таблица 38. Переменные функции input\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент списка |
| **Локальные параметры** | menu | int | Выбранный пункт меню |
| file\_name | char \* | Указатель на первый символ строки пути к файлу |
| nsp | OSHIP\* | Указатель на первый элемент введённого списка |

**7.34. Функция output\_menu**

**Назначение функции**  
Функция для создания меню вывода списка.  
  
**Прототип**  
void output\_menu(OSHIP \*ship,char\* title);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
output\_menu(ship\_res, "результатов поиска");  
Фактические параметры указаны в пункте 7.41  
  
**Описание переменных**  
Описание переменных представлено в таблице 39.

*Таблица 39. Переменные функции output\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент списка с результатами поиска |
| title | char\* | Указатель на первый элемент строки с назманием того, что выводим |
| **Локальные параметры** | menu | int | Выбранный пункт меню |
| file\_name | char \* | Указатель на первый символ строки пути к файлу |

**7.35. Функция remove\_menu**

**Назначение функции**  
Функция для создания меню удалений.  
  
**Прототип**  
OSHIP\* remove\_menu(OSHIP\* ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship = remove\_menu(ship);  
Фактические параметры указаны в пункте 7.1  
  
**Описание переменных**  
Описание переменных представлено в таблице 40.

*Таблица 40. Переменные функции remove\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные параметры** | menu | int | Выбранный пункт меню |
| file\_name | char \* | Указатель на первый символ строки пути к файлу |

**7.36. Функция help\_menu**

**Назначение функции**  
Функция для создания меню справки.  
  
**Прототип**  
void help\_menu();  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
help\_menu();  
  
**Описание переменных**  
Описание переменных представлено в таблице 41.

*Таблица 41. Переменные функции help\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Локальные параметры** | menu | int | Выбранный пункт меню |

**7.37. Функция O\_create\_copy**

**Назначение функции**  
Это функция для создания копии элемента списка.  
  
**Прототип**  
OSHIP \*O\_create\_copy(OSHIP\* ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на коированный элемент.  
  
**Пример вызова**  
new\_el = O\_create\_copy(ship);  
Фактические параметры указаны в пункте 7.38  
  
**Описание переменных**  
Описание переменных представлено в таблице 42.

*Таблица 42. Переменные функции O\_create\_copy*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные параметры** | new\_ship | OSHIP \* | Указатель на копию элемента |
| i | int | Вспомогательная переменная |
| len | int | Количество элементов в списке |

**7.38. Функция O\_find**

**Назначение функции**  
Это функция для поиска.  
  
**Прототип**  
OSHIP\* O\_find(FINDDATA\*find, OSHIP\*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship\_res = O\_find(&find, ship);  
Фактические параметры указаны в пункте 7.41  
  
**Описание переменных**  
Описание переменных представлено в таблице 43.

*Таблица 43. Переменные функции O\_find*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | find | FINDDATA\* | Указатель на структуру с параметрами поиска |
| ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные параметры** | ship\_res | OSHIP\* | Указатель на первый элемент списка с результатами поиска |
| link | OSHIP\* | Вспомогательный указатель |
| new\_el | OSHIP\* | Указатель на копированный элемент |

**7.39. Функция O\_insert**

**Назначение функции**  
Это функция для вставления элемента в отсортированный список.  
  
**Прототип**  
OSHIP \*O\_insert(OSHIP \*ship, OSHIP \*el, SORTDATA\*sort);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship\_sorted = O\_insert(ship\_sorted, link, sort);  
Фактические параметры указаны в пункте 7.40  
  
**Описание переменных**  
Описание переменных представлено в таблице 44.

*Таблица 44. Переменные функции O\_insert*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент отсортированного списка |
| el | OSHIP \* | Указатель на вставляемый элемент |
| sort | SORTDATA\* | Указатель на структуру с параметрами сортировки |
| **Локальные параметры** | l | OSHIP\* | Вспомогательный указатель |

**7.40. Функция O\_sort**

**Назначение функции**  
Это функция для сортировки.  
  
**Прототип**  
OSHIP \*O\_sort(OSHIP \*ship, SORTDATA \*sort);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship=O\_sort(ship, &sort);  
Фактические параметры указаны в пункте 7.41  
  
**Описание переменных**  
Описание переменных представлено в таблице 45.

*Таблица 45. Переменные функции O\_sort*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент списка |
| sort | SORTDATA \* | Указатель на структуру с параметрами сортировки |
| **Локальные параметры** | ship\_sorted | OSHIP \* | Указатель на первый элемент отсортированного списка |
| link | OSHIP \* | Указатель на вставляемый элемент |

**7.41. Функция calculate\_menu**

**Назначение функции**  
Это функция для создания меню обработки списка.  
  
**Прототип**  
OSHIP\* calculate\_menu(OSHIP\* ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship=calculate\_menu(ship);  
Фактические параметры указаны в пункте 7.1  
  
**Описание переменных**  
Описание переменных представлено в таблице 46.

*Таблица 46. Переменные функции calculate\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные параметры** | link | OSHIP\* | Вспомогательный указатель |
| ship\_res | OSHIP\* | Указатель на первый элемент списка с результатами поиска |
| menu | int | Выбранный пункт меню |

**7.42. Функция edit\_one**

**Назначение функции**  
Это функция для редактирования одного элемента.  
  
**Прототип**  
void edit\_one(OSHIP \*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
edit\_one(link);  
Фактические параметры указаны в пункте 7.43  
  
**Описание переменных**  
Описание переменных представлено в таблице 47.

*Таблица 47. Переменные функции edit\_one*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на редактируемый элемент |
| **Локальные параметры** | menu | int | Выбранный пункт меню |

**7.43. Функция charg\_menu**

**Назначение функции**  
Это функция для создания меню изменения элементов списка.  
  
**Прототип**  
void charg\_menu(OSHIP \*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
charg\_menu(ship);  
Фактические параметры указаны в пункте 7.46  
  
**Описание переменных**  
Описание переменных представлено в таблице 48.

*Таблица 48. Переменные функции charg\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент списка |
| **Локальные параметры** | link | OSHIP\* | Указатель на редактируемый элемент |
| len | int | Количество элементов в списке |
| n | int | Номер изменяемого элемента |

**7.44. Функция O\_del\_one**

**Назначение функции**  
Это функция для удаления одного элемента из списка.  
  
**Прототип**  
void O\_del\_one(OSHIP \*\*link);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Функция не возвращает значения.  
  
**Пример вызова**  
O\_del\_one(link\_l);  
Фактические параметры указаны в пункте 7.45  
  
**Описание переменных**  
Описание переменных представлено в таблице 49.

*Таблица 49. Переменные функции O\_del\_one*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | link | OSHIP \*\* | Указатель на ячейку памяти в которой хранится адрес удаляемого элемента |
| **Локальные параметры** | l | OSHIP \* | Вспомогательный указатель |

**7.45. Функция del\_menu**

**Назначение функции**  
Это функция для создания меню удаления элементов.  
  
**Прототип**  
OSHIP\* del\_menu(OSHIP\* ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship = del\_menu(ship);  
Фактические параметры указаны в пункте 7.46  
  
**Описание переменных**  
Описание переменных представлено в таблице 50.

*Таблица 50. Переменные функции del\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные параметры** | menu | int | Выбранный пункт меню |
| len | int | Количество элементов в списке |
| link | OSHIP \* | Вспомогательный указатель |
| link\_l | OSHIP \*\* | Указатель на ячейку памяти в которой хранится адрес удаляемого элемента |
| n | int | Номер первого удалемого элемента |
| m | int | Номер последнего удалемого элемента |
| i | int | Вспомогательная переменная |

**7.46. Функция edit\_menu**

**Назначение функции**  
Это функция для создания меню редактирования.  
  
**Прототип**  
OSHIP\* edit\_menu(OSHIP \*ship);  
Описание формальных параметров представлено ниже.  
  
**Возвращаемое значение**  
Эта функция возвращает указатель на первый элемент списка.  
  
**Пример вызова**  
ship = edit\_menu(ship);  
Фактические параметры указаны в пункте 7.1  
  
**Описание переменных**  
Описание переменных представлено в таблице 51.

*Таблица 51. Переменные функции edit\_menu*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Формальные параметры** | ship | OSHIP \* | Указатель на первый элемент списка |
| **Локальные параметры** | menu | int | Выбранный пункт меню |

**7.47. Функция my\_getch**

**Назначение функции**  
Функция для получения символа.   
  
**Прототип**  
char my\_getch();  
  
**Возвращаемое значение**  
Введённый символ.  
  
**Пример вызова**  
switch (ch = my\_getch())  
Фактические параметры указаны в пункте 7.5  
  
**Описание переменных**  
Описание переменных представлено в таблице 52.

*Таблица 52. Переменные функции my\_getch*

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя переменной** | **Тип переменной** | **Назначение переменной** |
| **Локальные параметры** | ch | char | Код нажатой кнопки. |

**8. Исходный код программы**

#include <conio.h>

#include <malloc.h>

#include <stdio.h>

#include <windows.h>

#include <locale.h>

typedef struct

{

char \*name;//Указатель на первый символ строки, содержащий имя кораблика

char \*type;//Указатель на первый символ строки, содержащий тип кораблика

int x; //Положение кораблика по x

int y; //Положение кораблика по y

int z; //Положение кораблика по z

int m; //Водоизмещение в мл

int speed; //Скорость в см/с

}DATA;

typedef struct ST1

{

DATA info;

struct ST1 \*link1;//Ссылка на следующий элемент

}OSHIP;

typedef struct

{

char s\_name; //Тип сортировки по имени(1-по возрастанию 2-по убыванию)

char s\_type; //Тип сортировки по типу(1-по возрастанию 2-по убыванию)

char s\_x; //Тип сортировки по x(1-по возрастанию 2-по убыванию)

char s\_y; //Тип сортировки по y(1-по возрастанию 2-по убыванию)

char s\_z; //Тип сортировки по z(1-по возрастанию 2-по убыванию)

char s\_m; //Тип сортировки по водоизмещению(1-по возрастанию 2-по убыванию)

char s\_speed;//Тип сортировки по скорости(1-по возрастанию 2-по убыванию)

char typ[7]; //Массив размера 7, в котором элемент означает критерий сортировки (1-имя

2-тип 3-x 4-y 6-m 5-speed 0-нет критерия)

} SORTDATA;

typedef struct

{

char \*name;//Указатель на первый символ строки, содержащий имя кораблика

char \*type;//Указатель на первый символ строки, содержащий тип кораблика

int x; //Положение кораблика по x

int y; //Положение кораблика по y

int z; //Положение кораблика по z

int m; //Положение кораблика по z

int speed; //Положение кораблика по z

char s\_name; //Тип поиска по имени(0-не важно 1-те, что больше 2-те, что меньше 3-

полное совпадение)

char s\_type; //Тип поиска по типу(0-не важно 1-те, что больше 2-те, что меньше 3-

полное совпадение)

char s\_x; //Тип поиска по x(0-не важно 1-те, что больше 2-те, что меньше 3-полное

совпадение)

char s\_y; //Тип поиска по y(0-не важно 1-те, что больше 2-те, что меньше 3-полное

совпадение)

char s\_z; //Тип поиска по z(0-не важно 1-те, что больше 2-те, что меньше 3-полное

совпадение)

char s\_m; //Тип поиска по водоизмещению(0-не важно 1-те, что больше 2-те, что

меньше 3-полное совпадение)

char s\_speed;//Тип поиска по скорости(0-не важно 1-те, что больше 2-те, что меньше 3-

полное совпадение)

} FINDDATA;

#define S\_input\_path\_help "Запрещено:\n 1. Вводить символы кириллицы\n 2. Вводить пробел в начале\n 3. Вводить несколько пробелов подряд\nВвод запрещённых символов игнорируется.\nПробел в конце пути будет игнорироваться.\nВы также можете добавить расширение \".txt\", для того, чтобы можно\nбыло открыть файл в блокноте.\n\nДля окончания ввода нажмите Enter\n"

#define S\_input\_nt\_help "Запрещено:\n 1. Вводить символы кириллицы\n 2. Вводить пробел в начале\n 3. Вводить несколько пробелов подряд\n 4. Вводить какие либо символы, кроме английских букв, пробела и\n круглых скобок\nВвод запрещённых символов будет игнорироваться.\nПробел в конце будет игнорироваться.\nВводимая строка должна состоять из одного или нескольких слов,\nпервое из которых должно начинаться с строчной буквы.\nВсе строчные буквы в слове после первой будут сделаны прописными.\nВводимая строка может содержать от 1 до 50 символов.\nПри достижении максимального количества символов ввод\nсимволов будет игнорироваться.\nДля окончания ввода нажмите Enter\n"

#define S\_input\_number "Разрешено вводить только цифры и '-'.\nОбязательно надо ввести число.\n"

#define S\_table\_fringe "|----------------------------------------------------|----------------------------------------------------|---------|---------|---------|-------------|--------|\n"

/\*

Описание: Функция для получения символа.

Возврат: Введённый символ.

\*/

char my\_getch()

{

char ch;//Код нажатой кнопки.

do

{

if ((ch = \_getch()) == '\r')

ch = '\n';

else

if (ch == '\0' || ch == -32)

{

if (kbhit())

\_getch();

}

else

if (ch == '\t')

ch = ' ';

} while (ch >= -128 && ch <= -17 || ch == '\0' || ch == -32);

return ch;

}

/\*

Описание: Функция для ввода целого числа.

Возврат: Введённое число.

\*/

int input\_number()

{

int number; //Введённое число

char \*s; //Указатель на первый символ исходной строки

int length = 1;//Длина строки

char ch; //Введённый символ

s = (char\*)malloc(sizeof(char));

do

{

switch (ch = my\_getch())//Получаем символ нажатой кнопки

{

case '\b':

if (length > 1)

{

printf("\b \b");

length--;

s = (char \*)realloc(s, sizeof(char)\*length);

}

break;

case '\n':

s[length - 1] = '\0';

break;

default:

if ((ch >= '0' && ch <= '9' || ch == '-' && length == 1) && length <= 7)

{

s = (char \*)realloc(s, sizeof(char)\*(length + 1));

printf("%c", ch);

s[length - 1] = ch;

length++;

}

break;

}

} while (ch != '\n' || length == 1 || length == 2 && \*s == '-');

number = atoi(s);

free(s);

printf("\n");

return number;

}

/\*

Описание: Функция для ввода ответа y/n.

Возврат: 1 - 'y', 0 - 'n'.

\*/

bool input\_yn()

{

char sy; //Введённый символ

char ch; //Нажатая клавиша

printf("Ваш выбор(Можно вводить только 'y' и 'n'): ");

sy = '\0';

do

{

switch (ch = my\_getch())//Получаем символ нажатой клавиши

{

case '\b':

if (sy != '\0')

{

printf("\b \b");

sy = '\0';

}

break;

case 'y':

case 'Y':

case 'n':

case 'N':

if (sy != '\0')

printf("\b \b");

printf("%c", sy = ch);

break;

}

} while (ch != '\n' || sy == '\0');

printf("\n");

return sy == 'y' || sy == 'Y';

}

/\*

Описание: Функция для проверки, является ли разрешённым символ.

Возврат: 1 - является, 0 - не является.

\*/

bool is\_allowed(char ch, char \*not\_allowed)

{

bool b;//Флажок совпадения символа с запрещённым

for (b = 0; !b && \*not\_allowed != '\0'; not\_allowed++)

b = ch == \*not\_allowed;

return !b;

}

/\*

Описание: Функция для ввода строки.

Возврат: Указатель на первый символ введённой строки.

\*/

char \*input\_string(int size,char \*not\_allowed)

{

char \*s; //Указатель на первый символ исходной строки

int length = 1;//Длина строки

char ch; //Введённый символ

s = NULL;

do

{

switch (ch = my\_getch())

{

case '\b':

if (length > 1)

{

printf("\b \b");//Стираем 1 символ

length--;

s = (char \*)realloc(s, sizeof(char)\*length);

}

break;

case '\n':

if (length > 1)

{

while (s[length - 2] == ' ')

{

length--;

s = (char \*)realloc(s, sizeof(char)\*length);

}

s[length - 1] = '\0';

}

break;

default:

if (is\_allowed(ch, not\_allowed) && length <= size && (ch != ' ' || length > 1 &&

s[length - 2] != ' '))

{

s = (char \*)realloc(s, sizeof(char)\*(length + 1));

printf("%c", ch);

s[length - 1] = ch;

length++;

}

break;

}

} while (ch != '\n' || length == 1);

printf("\n");

return s;

}

/\*

Описание: Функция для вычисления длины строки.

Возврат: Длина строки.

\*/

int str\_len(char\*s)

{

int i = 1;//Длина строки

while (\*s)

i++, s++;

return i;

}

/\*

Описание: Функция для проверки, является ли символ прописной буквой.

Возврат: 1 - является, 0 - не является.

\*/

bool is\_sm\_letter(char ch)

{

return ch >= 'a' && ch <= 'z';

}

/\*

Описание: Функция для проверки, является ли символ строчной буквой.

Возврат: 1 - является, 0 - не является.

\*/

bool is\_bi\_letter(char ch)

{

return ch >= 'A' && ch <= 'Z';

}

/\*

Описание: Функция для коррекции введённой строки.

Возврат: Функция не возвращает значения.

\*/

void correct\_name(char\* s)

{

if (s == NULL)

return;

if (\*s == 0)

return;

char ra = 'A' - 'a';//Разница в кодах прописных и строчных букв

bool is\_p\_let; //Флажок предыдущего символа(1-предыдущий символ буква 0-

предыдущий символ не буква)

if (is\_sm\_letter(\*s))

{

\*s += ra;//Делаем из прописной буквы строчную

is\_p\_let = 1;

}

else

is\_p\_let = is\_bi\_letter(\*s);

for (s = s + 1; \*s; s++)

if (is\_p\_let)

if (is\_bi\_letter(\*s))

\*s = \*s - ra;//Делаем из строчной буквы прописную

else

is\_p\_let = is\_sm\_letter(\*s);

else

is\_p\_let = is\_sm\_letter(\*s) || is\_bi\_letter(\*s);

}

/\*

Описание: Функция для ввода имени.

Возврат: Указатель на первый символ введённой строки.

\*/

char\* get\_nt(char \*title,char\* name)

{

char \*s; //Указатель на первый символ имени

bool exit;//Флажок выхода

exit = 0;

do

{

system("cls");

puts(title);

printf(S\_input\_nt\_help);

printf("Вводите %s:\n", name);

s = input\_string(50, "\x1b[]{}\\|\*/-+,.^%$#@!&\_=:;\"'1234567890`~");

correct\_name(s);

printf("Распознано как:\n%s\n", s);

puts("Желаете вести ещё раз?('y'-да 'n'-нет)");

if (input\_yn())

{

free(s);

s = NULL;

}

else

exit = 1;

} while (!exit);

return s;

}

/\*

Описание: Функция для ввода пути к файлу.

Возврат: Указатель на первый символ введённой строки.

\*/

char\* get\_path(char\* demo\_f)

{

char\* file\_name;//Указатель на первый символ строки, содержащей путь к файлу

system("cls");

if (demo\_f!=NULL)

printf("Желаете использовать демо-файл: \"%s\"?\n", demo\_f);

if (input\_yn()&& demo\_f!=NULL)

{

file\_name = (char\*)malloc(8\*sizeof(char));

strcpy(file\_name,"txt.txt");

}

else

{

printf(S\_input\_path\_help);

puts("Вводите путь к файлу(путь к файлу должен содержать хотя бы 1 символ,");

puts("но не более 70):");

file\_name = input\_string(70, "");

}

return file\_name;

}

/\*

Описание: Функция для ввода типа кораблика.

Возврат: Указатель на первый символ введённой строки.

\*/

char\* get\_type(char\* title)

{

char \*types[] = { "Unknown", "Pinnace", "Rowbarge",

"Galleon", "Shallop", "Bark", "Galliot", "Brigantine",

"Galley", "Hoy", "Ketch", "Ship", "Sloop", "Fireship",

"Ship of the Line", "Bomb Vessel", "Storeship", "Snow",

"Yacht", "Praam" };

//Массив указателей на первый символ строки

int i; //Вспомогательная переменная

char \*s; //Указатель на первый символ вводимой строки

char type;//Тип кораблика

char len; //Длина строки

puts("Выберите тип:");

for (i = 0; i < 20; i++)

printf("%i-%s\n", i + 1, types[i]);

puts("другое число-ввод типа вручную");

printf("Ваш выбор: ");

type = input\_number();

if (type <= 0 || type > 20)

{

system("cls");

s = get\_nt(title,"тип");//Ввод типа вручную

}

else

{

len = str\_len(types[type - 1]);

s = (char\*)malloc(sizeof(char)\*len);

for (i = 0; i < len; i++)

s[i] = types[type - 1][i];

}

return s;

}

/\*

Описание: Функция для ввода координат.

Возврат: Введённая координата.

\*/

int get\_coord(char\* title,char ch)

{

puts(title);

printf(S\_input\_number);

printf("Вводите положение по оси %c в сантиметрах (целое число от -999999 до

9999999):\n", ch);

return input\_number();

}

/\*

Описание: Это функция для ввода числа в заданных пределах.

Возврат: Эта функция возвращает введённое число.

\*/

int get\_sized(char\* title,int min, int max, char\* name)

{

int num;//Введённое число

do

{

puts(title);

printf(S\_input\_number);

printf("Вводите %s (целое число от %i до %i):\n", name, min, max);

num = input\_number();

if (num < min || num > max)

{

printf("Ошибка. Пожалуйста введите %s(от %i до %i)\n", name, min, max);

system("pause");

system("cls");

}

} while (num < min || num > max);

return num;

}

/\*

Описание: Функция получения полей структуры.

Возврат: Функция не возвращает значения.

\*/

void get\_struct\_info(DATA \*info)

{

system("cls");

info->name = get\_nt("Ввод данных кораблика","имя");

system("cls");

info->type = get\_type("Ввод данных кораблика");

system("cls");

info->x = get\_coord("Ввод данных кораблика",'x');

system("cls");

info->y = get\_coord("Ввод данных кораблика",'y');

system("cls");

info->z = get\_coord("Ввод данных кораблика",'z');

system("cls");

info->m = get\_sized("Ввод данных кораблика",10, 100000, "водоизмещение");

system("cls");

info->speed = get\_sized("Ввод данных кораблика", 0, 1000, "скорость в см/с");

}

/\*

Описание: Функция для ввода списка.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP \*O\_get\_structs()

{

OSHIP\* n\_sp; //Указатель на первый элемент нового списка

OSHIP \*n\_el; //Указатель на новый элемент

OSHIP \*last\_el; //Указатель на последний элемент списка

bool is\_end\_input;//Флажок выхода

n\_sp = (OSHIP\*)malloc(sizeof(OSHIP));

get\_struct\_info(&(n\_sp->info));

printf("Продолжить ввод?('y'-да 'n'-нет):\n");

if (input\_yn())

{

last\_el = n\_sp;

do

{

n\_el = (OSHIP\*)malloc(sizeof(OSHIP));

get\_struct\_info(&(n\_el->info));

last\_el->link1 = n\_el;

last\_el = n\_el;

printf("Продолжить ввод?('y'-да 'n'-нет):\n");

is\_end\_input = !input\_yn();

} while (!is\_end\_input);

last\_el->link1 = NULL;

}

else

n\_sp->link1 = NULL;

return n\_sp;

}

/\*

Описание: Функция для вывода информационных полей одной структуры.

Возврат: Функция не возвращает значения.

\*/

void O\_print\_struct(OSHIP \*ship)

{

printf("Имя кораблика: \"%s\"\n", ship->info.name);

printf("Тип кораблика: \"%s\"\n", ship->info.type);

printf("Положение кораблика (x,y,z) в сантиметрах: (%i,%i,%i)\n", ship->info.x,

ship->info.y, ship->info.z);

printf("Водоизмещение: %i\n", ship->info.m);

printf("Скорость в см/с: %i\n", ship->info.speed);

}

/\*

Описание: Функция для вывода списка.

Возврат: Функция не возвращает значения.

\*/

void O\_print(OSHIP \*ship)

{

int i;//Номер кораблика

puts("Вывод с первого элемента");

if (ship != NULL)

{

i = 0;

do

{

if (i > 0 && i % 3 == 0)

{

system("pause");

system("cls");

}

printf("Кораблик номер %i\n", i + 1);

O\_print\_struct(ship);

puts("");

ship = ship->link1;//Переходим к следующему символу

i++;

} while (ship != NULL);

puts("Вывод окончен");

}

else

puts("Картотека пуста");

}

/\*

Описание: Функция для освобождения памяти из-под всего списка.

Возврат: Эта функция возвращает NULL.

\*/

OSHIP\* O\_free\_all(OSHIP\* ship)

{

OSHIP \*link;//Указатель на элемент, из-под которого освобождается память

while (ship != NULL)

{

link = ship;//Сохраняем указатель на удаляемый элемент

ship = ship->link1;//Переходим к следующему символу

free(link->info.name);

link->info.name = NULL;

free(link->info.type);

link->info.type = NULL;

free(link);

link = NULL;

}

return NULL;

}

/\*

Описание: Это функция для подсчёта количества элементов в списке.

Возврат: Эта функция возвращает количество элементов в списке.

\*/

int O\_get\_len(OSHIP\* ship)

{

int i; //Количество элементов в списке

i = 0;

while (ship != NULL)

{

i++;

ship = ship->link1;

}

return i;

}

/\*

Описание: Функция для считывания из файла строки.

Возврат: Эта функция возвращает указатель на первый символ считанной строки.

\*/

char\* F\_gets(char end, int maxlen, FILE\* f)

{

if (feof(f))

return NULL;

char \*s; //Указатель на первый символ считываемой строки

bool exit;//Флажок выхода

int len; //Длина строки

s = NULL;

len = 0;

exit = 0;

do

{

s = (char\*)realloc(s, sizeof(char)\*(len + 1));

exit = !fscanf(f,"%c",s + len);

if (exit)

{

free(s);

s = NULL;

}

else

{

if (s[len] == end)

{

s[len] = '\0';

exit = 1;

}

len++;

}

} while (!exit);

return s;

}

/\*

Описание: Это функция для добавления списка после определённого элемента исходного списка.

Возврат: Функция не возвращает значения.

\*/

void O\_add\_after(OSHIP \*sp, OSHIP \*add\_sp)

{

if (sp != NULL)

{

OSHIP \*li;//Вспомогательный указатель

for (li = add\_sp; li->link1 != NULL; li = li->link1);

li->link1 = sp->link1;

sp->link1 = add\_sp;

}

}

/\*

Описание: Функция для проверки правильности данных в файле.

Возврат: Эта функция возвращает 1 - если данные правильны, 0 - если файл пуст, или его не существует, -1 - если данные в файле повреждены.

\*/

int test\_file(char\*file\_name)

{

FILE\* f; //Указатель на данные о файле

char\* s; //Указатель на первый символ строки

int i; //Вспомогательная переменная

if ((f = fopen(file\_name, "r")) == NULL)

return 0;

if (feof(f))

return 0;

do

{

s = F\_gets('\n', 50, f);

if (s == NULL)

{

fclose(f);

return -1;

}

if (s[0] == '\0')

if (feof(f))

{

free(s);

fclose(f);

return 1;

}

else

{

free(s);

fclose(f);

return -1;

}

s = F\_gets('\n', 20, f);

if (s == NULL || s[0]=='\0')

{

free(s);

fclose(f);

return -1;

}

if (fscanf(f, "%i %i %i %i %i", &i, &i, &i, &i, &i) != 5)

{

fclose(f);

return -1;

}

fscanf(f,"\n");

} while (!feof(f));

fclose(f);

return 1;

}

/\*

Описание: Функция для вывода информационных полей в файл в "красивом" виде.

Возврат: Эта функция возвращает 1 - если удалось записать, иначе 0.

\*/

bool save\_good\_placed(char\*file\_name, OSHIP\*ship)

{

if (ship == NULL)

{

puts("Нечего записывать");

return 0;

}

FILE\* f; //Указатель на данные о файле

if ((f = fopen(file\_name, "r")) != NULL)

fclose(f);

if (f == NULL)

{

f = fopen(file\_name, "w");

}

else

{

puts("Файл уже существует. Желаете перезаписать его?");

if (!input\_yn())

{

puts("Картотека не была записанна в файл");

return 0;

}

f = fopen(file\_name, "w");

}

if (f == NULL)

{

puts("Ошибка! Не удалось открыть файл");

return 0;

}

if (fputs(S\_table\_fringe, f) < 0)

{

puts("Во время записи произошла ошибка");

return fclose(f) \* 0;

}

if (fprintf(f, "|%52s|%52s|%9s|%9s|%9s|%13s|%8s|\n", "Имя кораблика", "Тип кораблика",

"x", "y", "z", "Водоизмещение", "Скорость") < 0)

{

puts("Во время записи произошла ошибка");

return fclose(f) \* 0;

}

if (fputs(S\_table\_fringe, f) < 0)

{

puts("Во время записи произошла ошибка");

return fclose(f) \* 0;

}

do

{

if( fprintf(f, "|%52s|%52s|%9i|%9i|%9i|%13i|%8i|\n", ship->info.name,

ship->info.type, ship->info.x, ship->info.y, ship->info.z, ship->info.m,

ship->info.speed) < 0)

{

puts("Во время записи произошла ошибка");

return fclose(f) \* 0;

}

if (fputs(S\_table\_fringe, f) < 0)

{

puts("Во время записи произошла ошибка");

return fclose(f) \* 0;

}

ship = ship->link1;

} while (ship != NULL);

fclose(f);

puts("Данные успешно записаны в файл");

return 1;

}

/\*

Описание: Функция для вывода информационных полей в файл.

Возврат: Эта функция возвращает 1 - если удалось записать, иначе 0.

\*/

bool save(char\*file\_name, OSHIP\*ship, bool is\_add)

{

if (ship == NULL)

{

puts("Нечего записывать");

return 0;

}

FILE\* f; //Указатель на данные о файле

OSHIP\* link; //Указатель на сохраняемый элемент

f ;

if ((f = fopen(file\_name, "r")) != NULL)

fclose(f);

if (!is\_add)

if (f == NULL)

{

f = fopen(file\_name, "w");

}

else

{

puts("Файл уже существует. Желаете перезаписать его?");

if (!input\_yn())

{

puts("Картотека не была записанна в файл");

return 0;

}

f = fopen(file\_name, "w");

}

else

if (f != NULL)

f = fopen(file\_name, "a");

if (f == NULL)

{

puts("Ошибка! Не удалось открыть файл");

return 0;

}

link = ship;

do

{

if (fprintf(f, "%s\n%s\n%i\n%i\n%i\n%i\n%i\n", link->info.name, link->info.type,

link->info.x, link->info.y, link->info.z, link->info.m, link->info.speed) < 0)

{

puts("Во время записи произошла ошибка");

return fclose(f) \* 0;

}

link = link->link1;

} while (link != NULL);

fclose(f);

puts("Данные успешно записаны в файл");

return 1;

}

/\*

Описание: Функция для ввода данных из файла.

Возврат: Эта функция возвращает указатель на первый элемент загруженного списка.

\*/

OSHIP \*load(char \*file\_name)

{

OSHIP\* ship;//Указатель на первый элемент списка

OSHIP\* el; //Вспомогательный указатель

OSHIP\* last;//Указатель на последний элемент списка

FILE \*f; //Указатель на данные о файле

bool err; //Флажок ошибки

bool end; //Флажок конца файла

char \*s; //Указатель на первый символ считанный элемент

f = fopen(file\_name, "r");

if (f == NULL)

return NULL;

err = 0;

end = 0;

ship = NULL;

do

{

s = F\_gets('\n', 50, f);

if (s != NULL)

{

if (s[0] != '\0')

{

el = (OSHIP\*)malloc(sizeof(OSHIP));

el->link1 = NULL;

el->info.type = NULL;

el->info.name = s;

if (ship != NULL)

{

O\_add\_after(last, el);

last = last->link1;

}

else

ship = last = el;

if ((s = F\_gets('\n', 50, f)) != NULL)

{

el->info.type = s;

err = fscanf(f, "%i %i %i %i %i", &(el->info.x), &(el->info.y),

&(el->info.z), &(el->info.m), &(el->info.speed)) != 5;

}

else

err = 1;

fscanf(f, "\n");

}

else

{

free(s);

end = 1;

}

}

else

if (feof(f))

end = 1;

else

err = 1;

} while (!err && !end);

if (err)

ship = O\_free\_all(ship);

fclose(f);

return ship;

}

/\*

Описание: Это функция для выбора типа сортировки.

Возврат: Функция не возвращает значения.

\*/

void select\_sort(SORTDATA \*sort)

{

bool b; //Вспомогательный флажок

int i; //Вспомогательная переменная

int len; //Количество критериев

int r; //Вспомогательная переменная

int o; //Вспомогательная переменная

do

{

puts("По скольким полям будет проводиться сортировка (от 1 до 7)?");

len = input\_number();

if (len < 1 || len > 7)

puts("Ошибка");

} while (len < 1 || len > 7);

puts("По какому критерию сортировать в первую очередь?");

do

{

puts("Варианты");

puts("1-имя");

puts("2-тип");

puts("3-x");

puts("4-y");

puts("5-z");

puts("6-водоизмещение");

puts("7-скорость");

r = input\_number();

if (r < 1 || r > 7)

puts("Ошибка");

} while (r< 1 || r > 7);

sort->typ[0] = r;

do

{

puts("Сортировать по данному критерию:");

puts("1-по возрастанию");

puts("2-по убыванию");

r = input\_number();

if (r < 1 || r > 7)

puts("Ошибка");

} while (r < 1 || r > 7);

switch (sort->typ[0])

{

case 1:

sort->s\_name = r;

break;

case 2:

sort->s\_type = r;

break;

case 3:

sort->s\_x = r;

break;

case 4:

sort->s\_y = r;

break;

case 5:

sort->s\_z = r;

break;

case 6:

sort->s\_m = r;

break;

case 7:

sort->s\_speed = r;

break;

}

for (i = 1; i < len; i++)

{

puts("По какому потом?");

do

{

do

{

puts("Варианты");

puts("1-имя");

puts("2-тип");

puts("3-x");

puts("4-y");

puts("5-z");

puts("6-водоизмещение");

puts("7-скорость");

r = input\_number();

if (r < 1 || r > 7)

{

puts("Нет такого пункта меню");

system("pause");

}

} while (r < 1 || r > 7);

b = 1;

for (o = 0; o < i && b; o++)

b = sort->typ[o] != r;

if (!b)

puts("Вы уже выбирали данное поле. Выберите теперь другое");

} while (!b);

sort->typ[i] = r;

do

{

puts("Сортировать по данному критерию:");

puts("1-по возрастанию");

puts("2-по убыванию");

r = input\_number();

if (r < 1 || r > 7)

puts("Ошибка");

} while (r < 1 || r > 7);

switch (sort->typ[i])

{

case 1:

sort->s\_name = r;

break;

case 2:

sort->s\_type = r;

break;

case 3:

sort->s\_x = r;

break;

case 4:

sort->s\_y = r;

break;

case 5:

sort->s\_z = r;

break;

case 6:

sort->s\_m = r;

break;

case 7:

sort->s\_speed = r;

break;

}

}

if (len < 7)

sort->typ[len] = 0;

}

/\*

Описание: Это функция для выбора типа поиска.

Возврат: Функция не возвращает значения.

\*/

void select\_find(FINDDATA \*find)

{

int i; //Вспомогательная переменная

int len; //Количество полей, по которым будет проводиться поиск

int r; //Вспомогательная переменная

char typ[7];//Массив размера 7 данных о сортировке

int o; //Вспомогательная переменная

for (i = 0; i < 7; i++)

typ[i] = 0;

find->s\_name = 0;

find->s\_type = 0;

find->s\_x = 0;

find->s\_y = 0;

find->s\_z = 0;

do

{

puts("По скольким полям будет проводиться поиск (от 1 до 7)?");

len = input\_number();

if (len < 1 || len > 7)

puts("Ошибка");

} while (len < 1 || len > 7);

i = 0;

do

{

do

{

do

{

puts("Выберите поле, по которому будет проводиться поиск");

puts("Варианты");

puts("1-имя");

puts("2-тип");

puts("3-x");

puts("4-y");

puts("5-z");

puts("6-водоизмещение");

puts("7-скорость");

r = input\_number();

if (r < 1 || r > 7)

puts("Ошибка");

} while (r < 1 || r > 7);

if (typ[r - 1] != 0)

puts("Вы уже выбирали данное поле. Выберите теперь другое");

} while (typ[r - 1] != 0);

typ[r - 1] = 1;

puts("Задайте значение для сравнения:");

switch (r)

{

case 1:

find->name = get\_nt("Ввод данных для сравнения", "имя кораблика");

break;

case 2:

find->type = get\_type("Ввод данных для сравнения");

break;

case 3:

find->x = get\_coord("Ввод данных для сравнения", 'x');

break;

case 4:

find->y = get\_coord("Ввод данных для сравнения", 'y');

break;

case 5:

find->z = get\_coord("Ввод данных для сравнения", 'z');

break;

case 6:

find->m = get\_sized("Ввод данных для сравнения", 10, 100000, "водоизмещение");

break;

case 7:

find->speed = get\_sized("Ввод данных для сравнения", 0, 1000, "скорость в см/с");

break;

}

do

{

puts("Задайте условие для сравнения при котором элемент подходит:");

puts("1-превышает заданное значение");

puts("2-меньше заданного значения");

puts("3-совпадает с заданным значением");

o = input\_number();

if (o < 1 || o > 3)

puts("Ошибка");

} while (o < 1 || o > 3);

switch (r)

{

case 1:

find->s\_name = o;

break;

case 2:

find->s\_type = o;

break;

case 3:

find->s\_x = o;

break;

case 4:

find->s\_y = o;

break;

case 5:

find->s\_z = o;

break;

case 6:

find->s\_m = o;

break;

case 7:

find->s\_speed = o;

break;

}

i++;

} while (i<len);

}

/\*

Описание: Это функция для сравнения двух элементов.

Возврат: Эта функция возвращает 1 если a меньше b, иначе 0.

\*/

bool a\_sma\_b(DATA a, DATA b, SORTDATA \* sort\_t)

{

char state;//Значение результата(1 - a < b, 0 - a == b, -1 - a > b)

int i; //Вспомогательная переменная

state = 0;

for (i = 0; i < 7 && sort\_t->typ[i] && !state; i++)

{

switch (sort\_t->typ[i])

{

case 1:

state = strcmp(a.name, b.name) > 0;

if (!state)

state = -(strcmp(a.name, b.name) < 0);

if (sort\_t->s\_name == 2)

state = -state;

break;

case 2:

state = strcmp(a.type, b.type)>0;

if (!state)

state = -(strcmp(a.type, b.type)<0);

if (sort\_t->s\_type == 2)

state = -state;

break;

case 3:

state = a.x > b.x;

if (!state)

state = -(a.x < b.x);

if (sort\_t->s\_x == 2)

state = -state;

break;

case 4:

state = a.y > b.y;

if (!state)

state = -(a.y < b.y);

if (sort\_t->s\_y == 2)

state = -state;

break;

case 5:

state = a.z > b.z;

if (!state)

state = -(a.z < b.z);

if (sort\_t->s\_z == 2)

state = -state;

break;

case 6:

state = a.m > b.m;

if (!state)

state = -(a.m < b.m);

if (sort\_t->s\_m == 2)

state = -state;

break;

case 7:

state = a.speed > b.speed;

if (!state)

state = -(a.speed < b.speed);

if (sort\_t->s\_speed == 2)

state = -state;

break;

}

}

return state == 1;

}

/\*

Описание: Функция для определения, подходит ли элемент критериям поиска.

Возврат: Эта функция возвращает 1 если подходит, иначе 0.

\*/

bool is\_need(DATA info, FINDDATA \*find)

{

switch (find->s\_name)

{

case 1:

if (strcmp(info.name, find->name) <= 0)

return 0;

break;

case 2:

if (strcmp(info.name, find->name) >= 0)

return 0;

break;

case 3:

if (strcmp(info.name, find->name) != 0)

return 0;

break;

}

switch (find->s\_type)

{

case 1:

if (strcmp(info.type, find->type) <= 0)

return 0;

break;

case 2:

if (strcmp(info.type, find->type) >= 0)

return 0;

break;

case 3:

if (strcmp(info.type, find->type) != 0)

return 0;

break;

}

switch (find->s\_x)

{

case 1:

if (info.x <= find->x)

return 0;

break;

case 2:

if (info.x >= find->x)

return 0;

break;

case 3:

if (info.x != find->x)

return 0;

break;

}

switch (find->s\_y)

{

case 1:

if (info.y <= find->y)

return 0;

break;

case 2:

if (info.y >= find->y)

return 0;

break;

case 3:

if (info.y != find->y)

return 0;

break;

}

switch (find->s\_z)

{

case 1:

if (info.z <= find->z)

return 0;

break;

case 2:

if (info.z >= find->z)

return 0;

break;

case 3:

if (info.z != find->z)

return 0;

break;

}

switch (find->s\_m)

{

case 1:

if (info.m <= find->m)

return 0;

break;

case 2:

if (info.m >= find->m)

return 0;

break;

case 3:

if (info.m != find->m)

return 0;

break;

}

switch (find->s\_speed)

{

case 1:

if (info.speed <= find->speed)

return 0;

break;

case 2:

if (info.speed >= find->speed)

return 0;

break;

case 3:

if (info.speed != find->speed)

return 0;

break;

}

return 1;

}

/\*

Описание: Это функция для добавления списка в начало исходного списка.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP\* O\_add\_frist(OSHIP \*sp, OSHIP \*add\_sp)

{

if (add\_sp != NULL)

{

OSHIP \*li;//Вспомогательный указатель

for (li = add\_sp; li->link1 != NULL; li = li->link1);

li->link1 = sp;

return add\_sp;

}

else

return sp;

}

/\*

Описание: Это функция для добавления списка в указаное место.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP \*O\_insert\_spis(OSHIP \*ship, OSHIP \*nsp)

{

int menu; //Выбранный пункт меню

int n; //Номер элемента

int len; //Количество элементов в списке

OSHIP \*link; //Указатель на вставляемый элемент

char\*file\_name;//Указатель на первый символ строки, содержащей путь к файлу

len = O\_get\_len(ship);

do

{

system("cls");

puts("Вставка");

puts("1-Вставить в начало картотеки");

puts("2-Вставить после n-ного элемента в картотеке");

puts("3-Вставить в конец картотеки, записанной в файл");

puts("0-Забыть данные, не вставляя их");

menu = input\_number();

if (menu<0 || menu>3)

{

puts("Нет такого пункта");

system("pause");

}

switch (menu)

{

case 1:

ship = O\_add\_frist(ship, nsp);

puts("Новые элементы успешно добавлены");

menu = 0;

break;

case 2:

if (ship != NULL)

{

do

{

printf("Вводите номер элемента, после которого будут вставленные\nэлементы

(от 1 до %i):", len);

n = input\_number();

if (n < 1 || n > len)

{

puts("Ошибка");

printf("Ведите число от 1 до %i\n",len);

system("pause");

}

} while (n < 1 || n > len);

for (link = ship; n > 1; link = link->link1, n--);

O\_add\_after(link, nsp);

puts("Новые элементы успешно добавлены");

menu = 0;

}

else

{

puts("Нет элементов в картокеке");

puts("Желаете добавить данные в начало картотеки?('y'-да 'n'-нет)");

if (input\_yn())

{

ship = nsp;

menu = 0;

puts("Новые элементы успешно добавлены");

}

}

break;

case 3:

file\_name = get\_path(NULL);

if (test\_file(file\_name) != -1)

if (save(file\_name, nsp,1))

{

nsp = O\_free\_all(nsp);

menu = 0;

}

else

system("pause");

else

{

puts("Выбранный вами файл повреждён или является файлом из другой программы");

system("pause");

}

free(file\_name);

file\_name = NULL;

break;

case 0:

nsp = O\_free\_all(nsp);

puts("Данные были успешно забыты");

break;

}

} while (menu);

return ship;

}

/\*

Описание: Функция для создания меню ввода.

Возврат: Эта функция возвращает указатель на первый элемент введённого списка.

\*/

OSHIP \*input\_menu(OSHIP \*ship)

{

int menu; //Выбранный пункт меню

char \*file\_name;//Указатель на первый символ строки пути к файлу

OSHIP\* nsp; //Указатель на первый элемент введённого списка

do

{

system("cls");

puts("Ввод данных в картотеку");

puts("1-с клавиатуры");

puts("2-из файла");

puts("0-возврат в надменю");

printf("Ваш выбор: ");

menu = input\_number();

switch (menu)

{

case 1:

nsp = O\_get\_structs();

puts("Данные успешно введены");

system("pause");

ship = O\_insert\_spis(ship, nsp);

nsp = NULL;

system("pause");

break;

case 2:

file\_name = get\_path("txt.txt");

nsp = load(file\_name);

if (nsp != NULL)

{

puts("Данные успешно загружены");

system("pause");

ship = O\_insert\_spis(ship, nsp);

nsp = NULL;

}

else

puts("Не удалось загрузить данные из файла");

free(file\_name);

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu);

return ship;

}

/\*

Описание: Функция для создания меню вывода списка.

Возврат: Функция не возвращает значения.

\*/

void output\_menu(OSHIP \*ship,char\* title)

{

if (ship == NULL)

{

puts("Картотека пуста редактировать нечего");

puts("Для ввода картотеки выберите: Главное меню->ввод данных в картотеку");

system("pause");

return;

}

int menu; //Выбранный пункт меню

char \*file\_name;//Указатель на первый символ строки пути к файлу

file\_name = (char\*)malloc(sizeof(char)\* 8);

strcpy(file\_name, "txt.txt");

do

{

system("cls");

printf("Вывод %s\n",title);

puts("1-на экран");

puts("2-в файл");

puts("3-дополнить существующий файл");

puts("0-возврат в надменю");

printf("Ваш выбор: ");

menu = input\_number();

switch (menu)

{

case 1:

system("cls");

O\_print(ship);

system("pause");

break;

case 2:

system("cls");

file\_name = get\_path("out.txt");

puts("Желаете записать файл в 'красивом' виде ?");

puts("При записи в 'красивом' виде файл потом будет невозможно прочитать");

puts("с помощю данной программы, но будет возможно прочитать с помощью");

puts("текстового редактора");

puts("(y-да n-нет):");

if (input\_yn())

save\_good\_placed(file\_name, ship);

else

save(file\_name, ship, 0);

free(file\_name);

system("pause");

break;

case 3:

system("cls");

file\_name = get\_path(NULL);

if (test\_file(file\_name) != -1)

save(file\_name, ship, 1);

else

puts("Выбранный вами файл повреждён или является файлом из другой программы");

free(file\_name);

file\_name = NULL;

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu);

}

/\*

Описание: Функция для создания меню удалений.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP\* remove\_menu(OSHIP\* ship)

{

int menu; //Выбранный пункт меню

char \*file\_name;//Указатель на первый символ строки пути к файлу

file\_name = (char\*)malloc(sizeof(char) \* 8);

strcpy(file\_name, "txt.txt");

do

{

system("cls");

puts("Удаление картотеки");

puts("1-картотеки из оперативной памяти");

puts("2-файла с картотекой");

puts("0-возврат в надменю");

printf("Ваш выбор: ");

menu = input\_number();

switch (menu)

{

case 1:

system("cls");

if (ship != NULL)

{

ship = O\_free\_all(ship);

puts("Удаление картотеки из оперативной памяти прошло успешно");

}

else

puts("Картотека пуста. Удалять из оперативной памяти нечего");

system("pause");

break;

case 2:

system("cls");

file\_name = get\_path("out.txt");

if (remove(file\_name) == 0)

puts("Файл с картотекой удалён");

else

puts("Не удалось удалить файл с картотекой");

free(file\_name);

file\_name = NULL;

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu);

return ship;

}

/\*

Описание: Функция для создания меню справки.

Возврат: Функция не возвращает значения.

\*/

void help\_menu()

{

int menu; //Выбранный пункт меню

do

{

system("cls");

puts("Справка");

puts("1-по вводу");

puts("2-по выводу");

puts("3-по структуре данных");

puts("4-ограничения");

puts("0-возврат в надменю");

printf("Ваш выбор: ");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

system("cls");

puts("Вывести пустую картотеку невозможно");

puts("При вводе имени и типа");

printf(S\_input\_nt\_help);

puts("");

puts("При вводе координат");

puts(S\_input\_number);

system("pause");

break;

case 2:

system("cls");

puts("При выводе на экран:");

puts(" Элементы выводятся последовательно");

puts(" Элементы выводятся парциально максимум по 3 шт. за раз");

puts("");

puts("При выводе в файл:");

puts(" Записывается каждый элемент списка в указанный вами файл");

puts(" Если в файле есть какая-то информация, то предлагается");

puts(" перезаписать его, или не перезаписывать");

puts(" Если файла не существует, то создаётся пустой файл и в него");

puts(" записывается список.");

puts(" Файл можно записать в 'красивом' и в обычном виде.");

puts(" При выводе в файл в 'красивом' виде, прочитать его с помощью");

puts(" этой программы будет невозможно, но будет возможно прочитать");

puts(" с помощью блокнота.");

puts(" При записи в обычном виде файл можно будет прочитать только");

puts(" с помощью данной программы.");

puts("При добавлении данных в файл:");

puts(" Данные в файле будут проверены на правильность.");

puts(" Данные будут добавлены в файл только если данные");

puts(" в файле правильны.");

puts(" Добавить данные в пустой файл невозможно.");

system("pause");

break;

case 3:

system("cls");

puts("Это программа для создания, загрузки, сохранения картотеки");

puts("Один элемент картотеки содержит данные об одном кораблике, а именно:");

puts(" 1. Имя кораблика - строка с максимальной длиной 50 символов");

puts(" 2. Тип кораблика - строка с максимальной длиной 50 символов");

puts(" 3. Положение кораблика по оси x");

puts(" 4. Положение кораблика по оси y");

puts(" 5. Положение кораблика по оси z");

puts(" 6. Водоизмещение");

puts(" 7. Скорость кораблика");

puts("Начало координат вы выбираете сами");

system("pause");

break;

case 4:

puts("Вывести пустую картотеку невозможно");

puts("Символы кириллицы вводить запрещено");

puts("Имя кораблика может максимально иметь 50 символов и состоит только из:");

puts(" -английских букв");

puts(" -пробелов");

puts(" -круглых скобок");

puts("Имя кораблика должно начинаться с большой буквы");

puts("Тип кораблика может максимально иметь 50 символов и состоит только из:");

puts(" -английских букв");

puts(" -пробелов");

puts(" -круглых скобок");

puts("Тип кораблика должен начинаться с большой буквы");

puts("Водоизмещение кораблика может быть только в мл и только целым числом от");

puts("10 до 100000");

puts("Скорость кораблика может быть только в см/с и только целым числом от");

puts("0 до 1000");

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu);

}

/\*

Описание: Это функция для создания копии элемента списка.

Возврат: Эта функция возвращает указатель на коированный элемент.

\*/

OSHIP \*O\_create\_copy(OSHIP\* ship)

{

OSHIP \*new\_ship;//Указатель на копию элемента

int i; //Вспомогательная переменная

int len; //Количество элементов в списке

if (ship != NULL)

{

new\_ship = (OSHIP\*)malloc(sizeof(OSHIP));

new\_ship->info.x = ship->info.x;

new\_ship->info.y = ship->info.y;

new\_ship->info.z = ship->info.z;

new\_ship->info.m = ship->info.m;

new\_ship->info.speed = ship->info.speed;

len = str\_len(ship->info.name);

new\_ship->info.name = (char\*)malloc(len\*sizeof(char));

for (i = 0; i < len; i++)

new\_ship->info.name[i] = ship->info.name[i];

len = str\_len(ship->info.type);

new\_ship->info.type = (char\*)malloc(len\*sizeof(char));

for (i = 0; i < len; i++)

new\_ship->info.type[i] = ship->info.type[i];

new\_ship->link1 = NULL;

}

else

new\_ship = NULL;

return new\_ship;

}

/\*

Описание: Это функция для поиска.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP\* O\_find(FINDDATA\*find, OSHIP\*ship)

{

if (ship == NULL)

{

puts("Нет элементов в картотеке");

return NULL;

}

OSHIP\* ship\_res;//Указатель на первый элемент списка с результатами поиска

OSHIP\* link; //Вспомогательный указатель

OSHIP\* new\_el; //Указатель на копированный элемент

ship\_res = NULL;

do

{

if (is\_need(ship->info, find))

{

new\_el = O\_create\_copy(ship);

if (ship\_res == NULL)

ship\_res = new\_el;

else

link->link1 = new\_el;

link = new\_el;

}

ship = ship->link1;

} while (ship != NULL);

if (ship\_res != NULL)

{

link->link1 = NULL;

puts("Поиск завершён");

}

else

puts("Не найдено элементов, удовлетворяющих заданным критериям");

return ship\_res;

}

/\*

Описание: Это функция для вставления элемента в отсортированный список.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP \*O\_insert(OSHIP \*ship, OSHIP \*el, SORTDATA\*sort)

{

if (ship != NULL)

if (!a\_sma\_b(ship->info, el->info, sort))

{

OSHIP\* l;//Вспомогательный указатель

l = ship;

while (ship->link1 != NULL)

{

if (a\_sma\_b(ship->link1->info, el->info, sort))

{

O\_add\_after(ship, el);

return l;

}

ship = ship->link1;

}

ship->link1 = el;

return l;

}

else

{

el->link1 = ship;

return el;

}

else

{

el->link1 = NULL;

return el;

}

}

/\*

Описание: Это функция для сортировки.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP \*O\_sort(OSHIP \*ship, SORTDATA \*sort)

{

if (ship == NULL)

{

puts("Нечего сортировать");

return NULL;

}

OSHIP \*ship\_sorted;//Указатель на первый элемент отсортированного списка

OSHIP \*link; //Указатель на вставляемый элемент

ship\_sorted = NULL;

do

{

link = ship;

ship = ship->link1;

link->link1 = NULL;

ship\_sorted = O\_insert(ship\_sorted, link, sort);

} while (ship != NULL);

puts("Сортировка окончена");

puts("Для вывода отсортированой картотеки выберите:");

puts("Главное меню->вывод картотеки");

return ship\_sorted;

}

/\*

Описание: Это функция для создания меню обработки списка.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP\* calculate\_menu(OSHIP\* ship)

{

if (ship == NULL)

{

puts("Картотека пуста редактировать нечего");

puts("Для ввода картотеки выберите: Главное меню->ввод данных в картотеку");

system("pause");

return ship;

}

SORTDATA sort; //Параметры сортировки

FINDDATA find; //Параметры поиска

OSHIP\*link; //Вспомогательный указатель

OSHIP\*ship\_res;//Указатель на первый элемент списка с результатами поиска

int menu; //Выбранный пункт меню

find.name = NULL;

find.type = NULL;

do

{

system("cls");

puts("Обработка");

puts("1-сортировка элементов картотеки");

puts("2-поиск");

puts("0-возврат в надменю");

printf("Ваш выбор: ");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

select\_sort(&sort);

ship=O\_sort(ship, &sort);

system("pause");

break;

case 2:

free(find.name);

find.name = NULL;

free(find.type);

find.type = NULL;

select\_find(&find);

ship\_res = O\_find(&find, ship);

system("pause");

if (ship\_res!=NULL)

{

output\_menu(ship\_res, "результатов поиска");

ship\_res = O\_free\_all(ship\_res);

}

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu);

free(find.name);

find.name = NULL;

free(find.type);

find.type = NULL;

return ship;

}

/\*

Описание: Это функция для редактирования одного элемента.

Возврат: Функция не возвращает значения.

\*/

void edit\_one(OSHIP \*ship)

{

int menu;//Выбранный пункт меню

do

{

system("cls");

puts("Редактируемый элемент:");

O\_print\_struct(ship);

puts("Меню редактирования");

puts("1-редактировать поле имя");

puts("2-редактировать поле тип");

puts("3-редактировать поле x");

puts("4-редактировать поле y");

puts("5-редактировать поле z");

puts("6-редактировать поле водоизмещение");

puts("7-редактировать поле скорость");

puts("0-выйти в надменю");

puts("Ваш выбор:");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

free(ship->info.name);

ship->info.name = NULL;

ship->info.name = get\_nt("Ввод данных для сравнения", "имя");

puts("Поле имя успешно отредактировано");

system("pause");

break;

case 2:

free(ship->info.type);

ship->info.type = NULL;

ship->info.type = get\_type("Ввод данных для сравнения");

puts("Поле тип успешно отредактировано");

system("pause");

break;

case 3:

ship->info.x = get\_coord("Ввод данных для сравнения", 'x');

puts("Поле x успешно отредактировано");

system("pause");

break;

case 4:

ship->info.y = get\_coord("Ввод данных для сравнения", 'y');

puts("Поле y успешно отредактировано");

system("pause");

break;

case 5:

ship->info.z = get\_coord("Ввод данных для сравнения", 'z');

puts("Поле z успешно отредактировано");

system("pause");

break;

case 6:

ship->info.m = get\_sized("Ввод данных для сравнения", 10, 100000,

"водоизмещение");

puts("Поле воодоизмещение успешно отредактировано");

system("pause");

break;

case 7:

ship->info.speed = get\_sized("Ввод данных для сравнения", 0, 1000,

"скорость в см/с");

puts("Поле скорость успешно отредактировано");

system("pause");

break;

case 0:

break;

}

} while (menu);

}

/\*

Описание: Это функция для создания меню изменения элементов списка.

Возврат: Функция не возвращает значения.

\*/

void charg\_menu(OSHIP \*ship)

{

if (ship == NULL)

{

puts("Картотека пуста обрабатывать нечего");

puts("Для ввода картотеки выберите: Главное меню->ввод данных в картотеку");

system("pause");

return;

}

OSHIP\* link; //Указатель на редактируемый элемент

int len = O\_get\_len(ship);//Количество элементов в списке

int n; //Номер изменяемого элемента

if (len != 0)

do

{

puts(S\_input\_number);

printf("Вводите номер редактируемого элемента от 1 до %i(0-возврат в

надменю):\n",len);

n = input\_number();

if (n < 0 || n > len)

{

puts("Ошибка! Вы ввели число вне заданых значений");

system("pause");

}

else

{

if (n > 0)

{

link = ship;

for (; n > 1; n--,link=link->link1);

edit\_one(link);

}

}

} while (n!=0);

}

/\*

Описание: Это функция для удаления одного элемента из списка.

Возврат: Функция не возвращает значения.

\*/

void O\_del\_one(OSHIP \*\*link)

{

if (\*link != NULL)

{

OSHIP \*l;//Вспомогательный указатель

l = \*link;

\*link = (\*link)->link1;

free(l->info.name);

free(l->info.type);

free(l);

}

}

/\*

Описание: Это функция для создания меню удаления элементов.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP\* del\_menu(OSHIP\* ship)

{

int menu; //Выбранный пункт меню

int len = O\_get\_len(ship);//Количество элементов в списке

OSHIP \*link; //Вспомогательный указатель

OSHIP \*\*link\_l; //Указатель на ячейку памяти в которой хранится адрес

удаляемого элемента

int n; //Номер первого удалемого элемента

int m; //Номер последнего удалемого элемента

int i; //Вспомогательная переменная

do

{

if (ship != NULL)

{

puts("Удаление элементов картотеки");

puts("1-удалить элементы с n до m");

puts("2-удалить первый элемент");

puts("3-удалить последний элемент");

puts("0-выйти в надменю");

puts("Ваш выбор:");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

n = get\_sized("Удаление элементов картотеки с n до m", 1, len, "n");

m = get\_sized("Удаление элементов картотеки с n до m", n, len, "m");

for (i = 2, link = ship; i < n; i++, link = link->link1);

if (n == 1)

link\_l = &ship;

else

link\_l = &link->link1;

i = m - n + 1;

do

{

O\_del\_one(link\_l);

i--;

} while (i > 0);

puts("Элементы успешно удалены");

system("pause");

len = O\_get\_len(ship);

break;

case 2:

O\_del\_one(&ship);

puts("Элемент успешно удален");

system("pause");

len--;

break;

case 3:

link\_l = &ship;

for (; (\*link\_l)->link1 != NULL; link\_l = &(\*link\_l)->link1);

O\_del\_one(link\_l);

puts("Элемент успешно удален");

system("pause");

len--;

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

}

else

{

puts("В картотеке больше нет элементов.");

puts("Сейчас вы вернётесь в главное меню");

system("pause");

menu = 0;

}

} while (menu);

return ship;

}

/\*

Описание: Это функция для создания меню редактирования.

Возврат: Эта функция возвращает указатель на первый элемент списка.

\*/

OSHIP\* edit\_menu(OSHIP \*ship)

{

if (ship == NULL)

{

puts("Картотека пуста редактировать нечего");

puts("Для ввода картотеки выберите: Главное меню->ввод данных в картотеку");

system("pause");

return ship;

}

int menu; //Выбранный пункт меню

do

{

system("cls");

puts("Редактирование картотеки");

puts("1-изменеине одного элемента");

puts("2-удаление элементов");

puts("0-возврат в надменю");

printf("Ваш выбор: ");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

charg\_menu(ship);

break;

case 2:

ship = del\_menu(ship);

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu&&ship!=NULL);

return ship;

}

int main()

{

setlocale(0, "RU");

int menu; //Выбранный пункт меню

OSHIP \*ship; //Указатель на первый элемент списка

//Изменение размера шрифта

HANDLE hCon = CreateConsoleScreenBuffer(GENERIC\_READ | GENERIC\_WRITE, 0, NULL, CONSOLE\_TEXTMODE\_BUFFER, NULL);

if (hCon != INVALID\_HANDLE\_VALUE) {

CONSOLE\_FONT\_INFOEX cfi;

cfi.cbSize = sizeof(CONSOLE\_FONT\_INFOEX);

cfi.nFont = 0;

cfi.dwFontSize.X = 0;

cfi.dwFontSize.Y = 24;

cfi.FontFamily = FF\_DONTCARE;

cfi.FontWeight = 400;

wcscpy(cfi.FaceName, L"Lucida Console");

SetCurrentConsoleFontEx(hCon, FALSE, &cfi);

}

puts(" II II II II ");

puts(" I II I II I II I II ");

puts(" I II I II I II I II ");

puts(" I II I II I II I II ");

puts(" I II I II I II I II ");

puts(" I II I II I II I II ");

puts(" IIIIIIIIIIIIIIIIIIIIIIIIIIIII IIIIIIIIIIIIIIIIIIIIIIIIIIIII ");

puts(" I I I I ");

puts(" III III III III ");

puts(" IIIII IIIII IIIII IIIII ");

puts(" IIIIIIIII IIIIIIIII ");

puts("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

puts("Вас приветствует программа для работы с картотекой.");

puts("Предметная область кораблики.");

puts("Один элемент картотеки содержит данные об одном кораблике, а именно:");

puts(" 1. Имя кораблика - строка с максимальной длиной 50 символов");

puts(" 2. Тип кораблика - строка с максимальной длиной 50 символов");

puts(" 3. Положение кораблика по оси x");

puts(" 4. Положение кораблика по оси y");

puts(" 5. Положение кораблика по оси z");

puts(" 6. Водоизмещение");

puts(" 7. Скорость кораблика");

system("pause");

ship = NULL;

do

{

system("cls");

puts("Главное меню");

puts("1-ввод данных в картотеку");

puts("2-вывод картотеки");

puts("3-обработка картотеки");

puts("4-редактирование картотеки");

puts("5-удаление картотеки");

puts("6-справка");

puts("0-выход");

printf("Ваш выбор: ");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

ship = input\_menu(ship);

break;

case 2:

output\_menu(ship,"картотеки");

break;

case 3:

ship=calculate\_menu(ship);

break;

case 4:

ship = edit\_menu(ship);

break;

case 5:

ship = remove\_menu(ship);

break;

case 6:

help\_menu();

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu);

ship = O\_free\_all(ship);

}

**9. Тесты**

**10. Инструкция пользователя**

**11. Результат**

При выполнении программы получены результаты, совпадающие с контрольными примерами, представленными в таблице 1. Ошибки не обнаружены.

**Вывод**

В результате выполнения курсовой работы были получены практические навыки по работе с электронными картотеками на языке C/C++.