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**

**Отчёт по лабораторной работе №2**

по дисциплине «Программирование»

на тему

«Списки С/C++»

Выполнил студент гр.5307 Самоуков Н.В.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"Выполнено" "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Санкт-Петербург**

**2015 г.**

Оглавление

[Цель работы 5](#_Toc448700873)

[1.Задание 5](#_Toc448700874)

[2.Уточнение задания 5](#_Toc448700875)

[3.Контрольные примеры 6](#_Toc448700876)

[4.Описание структур 7](#_Toc448700877)

[4.1 Описание полей структуры SORTDATA 7](#_Toc448700878)

[4.2 Описание полей структуры FINDDATA 8](#_Toc448700879)

[4.3 Описание полей структуры DATA 8](#_Toc448700880)

[4.4 Описание полей структуры OSHIP 9](#_Toc448700881)

[4.5 Описание полей структуры DSHIP 9](#_Toc448700882)

[4.6 Описание полей структуры CSHIP 9](#_Toc448700883)

[5.Описание переменных главной функции 10](#_Toc448700884)

[6.Краткое описание алгоритма 10](#_Toc448700885)

[7.Описание функций 11](#_Toc448700886)

[7.1 Функция my\_getch 11](#_Toc448700887)

[7.2 Функция input\_number 11](#_Toc448700888)

[7.3 Функция input\_yn 12](#_Toc448700889)

[7.4 Функция input\_string 13](#_Toc448700890)

[7.5 Функция str\_len 13](#_Toc448700891)

[7.6 Функция get\_type 14](#_Toc448700892)

[7.7 Функция get\_name 15](#_Toc448700893)

[7.8 Функция get\_coord 15](#_Toc448700894)

[7.9 Функция get\_struct\_info 16](#_Toc448700895)

[7.10 Функция select\_sort 17](#_Toc448700896)

[7.11 Функция select\_find 17](#_Toc448700897)

[7.12 Функция a\_sma\_b 18](#_Toc448700898)

[7.13 Функция is\_need 19](#_Toc448700899)

[7.14 Функция O\_get\_structs 20](#_Toc448700900)

[7.15 Функция O\_add\_frist 20](#_Toc448700901)

[7.16 Функция O\_add\_last 21](#_Toc448700902)

[7.17 Функция O\_add\_after 22](#_Toc448700903)

[7.18 Функция O\_add 23](#_Toc448700904)

[7.19 Функция O\_del\_one 23](#_Toc448700905)

[7.20 Функция O\_del 24](#_Toc448700906)

[7.21 Функция O\_create\_copy 25](#_Toc448700907)

[7.22 Функция O\_find 26](#_Toc448700908)

[7.23 Функция O\_insert 26](#_Toc448700909)

[7.24 Функция O\_sort 27](#_Toc448700910)

[7.25 Функция O\_print\_struct 28](#_Toc448700911)

[7.26 Функция O\_print 29](#_Toc448700912)

[7.27 Функция O\_free\_all 29](#_Toc448700913)

[7.28 Функция O\_menu 30](#_Toc448700914)

[7.29 Функция D\_get\_structs 31](#_Toc448700915)

[7.30 Функция D\_add\_frist 31](#_Toc448700916)

[7.31 Функция D\_add\_last 32](#_Toc448700917)

[7.32 Функция D\_add\_after 33](#_Toc448700918)

[7.33 Функция D\_add 33](#_Toc448700919)

[7.34 Функция D\_del\_one 34](#_Toc448700920)

[7.35 Функция D\_del 35](#_Toc448700921)

[7.36 Функция D\_create\_copy 35](#_Toc448700922)

[7.37 Функция D\_find 36](#_Toc448700923)

[7.38 Функция D\_insert 37](#_Toc448700924)

[7.39 Функция D\_sort 38](#_Toc448700925)

[7.40 Функция D\_print\_struct 38](#_Toc448700926)

[7.41 Функция D\_print 39](#_Toc448700927)

[7.42 Функция D\_free\_all 40](#_Toc448700928)

[7.43 Функция D\_menu 40](#_Toc448700929)

[7.44 Функция C\_add\_frist 41](#_Toc448700930)

[7.45 Функция C\_add\_last 42](#_Toc448700931)

[7.46 Функция C\_add\_after 43](#_Toc448700932)

[7.47 Функция C\_get\_structs 43](#_Toc448700933)

[7.48 Функция C\_get\_len 44](#_Toc448700934)

[7.49 Функция C\_add 45](#_Toc448700935)

[7.50 Функция C\_del\_one 46](#_Toc448700936)

[7.51 Функция C\_del 46](#_Toc448700937)

[7.52 Функция C\_print\_struct 47](#_Toc448700938)

[7.53 Функция C\_print 48](#_Toc448700939)

[7.54 Функция C\_free\_all 48](#_Toc448700940)

[7.55 Функция C\_create\_copy 49](#_Toc448700941)

[7.56 Функция C\_insert 50](#_Toc448700942)

[7.57 Функция C\_sort 51](#_Toc448700943)

[7.58 Функция C\_find 51](#_Toc448700944)

[7.59 Функция C\_menu 52](#_Toc448700945)

[8.Текст программы 53](#_Toc448700946)

[9.Результат 90](#_Toc448700947)

[Вывод 90](#_Toc448700948)

# Цель работы

Получить практические навыки использования списков в языке С/С++.

# 

# 1.Задание

Разработать алгоритм и написать программу, выполняющую многократно по желанию пользователя ввод, обработку, вывод для односвязных, двухсвязных и двухсвязных кольцевых списков.

# 

# 2.Уточнение задания

Есть 2 типа обработки: сортировка и поиск. Обработанный массив состоит из элементов исходного массива, удовлетворяющих задаваемым критериям. Сортировка проводится по пяти полям, задаваемым пользователем.

# 

# 3.Контрольные примеры

Контрольные примеры представлены в таблице 1. Контрольные примеры.

Таблица 1. Контрольные примеры

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Ожидаемый результат |
| 1 | 1)ship={"Ship1","boat",7,7,7}  2)ship={"Ship2","steelship",4,-5,10}  3)ship={"Ship3","boat",6,60,5}  Поиск  Имя:"Ship1"  Тип критерия: совпадает | 1)ship\_result={"Ship1","boat",7,7,7} |
| 2 | 1)ship={"Ship1","boat",7,7,7}  Поиск  Имя:"Ship1"  Тип критерия: совпадает | ship\_result=NULL |
| 3 | 1)ship={"Ship1","boat",7,7,7}  2)ship={"Ship2","steelship",4,-5,10}  3)ship={"Ship3","boat",6,60,5}  Поиск  Имя:"Ship1"  Тип критерия: совпадает | 1)ship\_result ={"Ship1","boat",7,7,7}  2)ship\_result ={"Ship3","boat",6,60,5} |
| 4 | 1)ship={"Ship 2","boat",7,7,7}  2)ship={"Ship 1","steelship",4,-5,10}  3)ship={"Ship 4","ship",6,60,5}  4)ship={"Ship 3","boat",-66,9,4}  5)ship={"Ship 5","steelship",4,-5,10}  6)ship={"Ship 8","boat",-1,-1,5}  7)ship={"Ship 6","boat",7,0,56}  8)ship={"Ship 7","boat",0,5,42}  Сортировка  1)Имя(по возрастанию) | 1)ship={"Ship 1","steelship",4,-5,10}  2)ship={"Ship 2","boat",7,7,7}  3)ship={"Ship 3","boat",-66,9,4}  4)ship={"Ship 4","ship",6,60,5}  5)ship={"Ship 5","steelship",4,-5,10}  6)ship={"Ship 6","boat",7,0,56}  7)ship={"Ship 7","boat",0,5,42}  8)ship={"Ship 8","boat",-1,-1,5} |
| 5 | 1)ship={"Ship 2","boat",7,7,7}  2)ship={"Ship 1","steelship",4,-5,10}  3)ship={"Ship 4","ship",6,60,5}  4)ship={"Ship 3","boat",-66,9,4}  5)ship={"Ship 5","steelship",4,-5,10}  6)ship={"Ship 8","boat",-1,-1,5}  7)ship={"Ship 6","boat",7,0,56}  8)ship={"Ship 7","boat",0,5,42}  Сортировка  1)x(по убыванию) | 1)ship={"Ship 2","boat",7,7,7}  2)ship={"Ship 6","boat",7,0,56}  3)ship={"Ship 4","ship",6,60,5}  4)ship={"Ship 1","steelship",4,-5,10}  5)ship={"Ship 5","steelship",4,-5,10}  6)ship={"Ship 7","boat",0,5,42}  7)ship={"Ship 8","boat",-1,-1,5}  8)ship={"Ship 3","boat",-66,9,4} |
| 6 | 1)ship={"Ship 2","boat",7,7,7}  2)ship={"Ship 1","steelship",4,-5,10}  3)ship={"Ship 4","ship",6,60,5}  4)ship={"Ship 3","boat",-66,9,4}  5)ship={"Ship 5","steelship",4,-5,10}  6)ship={"Ship 8","boat",-1,-1,5}  7)ship={"Ship 6","boat",7,0,56}  8)ship={"Ship 7","boat",0,5,42}  Сортировка  1)x(по убыванию)  2)Имя(по убыванию) | 1)ship={"Ship 6","boat",7,0,56}  2)ship={"Ship 2","boat",7,7,7}  3)ship={"Ship 4","ship",6,60,5}  4)ship={"Ship 5","steelship",4,-5,10}  5)ship={"Ship 1","steelship",4,-5,10}  6)ship={"Ship 7","boat",0,5,42}  7)ship={"Ship 8","boat",-1,-1,5}  8)ship={"Ship 3","boat",-66,9,4} |

# 4.Описание структур

# 4.1 Описание полей структуры SORTDATA

Описание полей структуры SORTDATA ,приведено в таблице 2 Описание полей структуры SORTDATA.

Таблица 2. 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) |

# 4.2 Описание полей структуры FINDDATA

Описание полей структуры FINDDATA ,приведено в таблице 3 Описание полей структуры FINDDATA.

Таблица 3. 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) |

# 4.3 Описание полей структуры DATA

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

Таблица 4. 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 |

# 4.4 Описание полей структуры OSHIP

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

Таблица 5. Описание полей структуры OSHIP.

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

# 4.5 Описание полей структуры DSHIP

Описание полей структуры DSHIP ,приведено в таблице 6 Описание полей структуры DSHIP.

Таблица 6. Описание полей структуры DSHIP.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| info | DATA | Данные о кораблике |
| link1 | OSHIP \* | Указатель на следующий элемент |
| link2 | OSHIP \* | Указатель на предыдущий элемент |

# 4.6 Описание полей структуры CSHIP

Описание полей структуры CSHIP ,приведено в таблице 7 Описание полей структуры CSHIP.

Таблица 7. Описание полей структуры CSHIP.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| info | DATA | Данные о кораблике |
| link1 | OSHIP \* | Указатель на следующий элемент |
| link2 | OSHIP \* | Указатель на предыдущий элемент |

# 5.Описание переменных главной функции

Переменные, используемые в главной функции, приведены в таблице 5 Переменные, используемые в главной функции.

Таблица 5. Переменные, используемые в главной функции

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| menu | int | Выбранный пункт меню |

# 6.Краткое описание алгоритма

Шаг 1. Вывести заставку.

Шаг 2. Очистить экран.

Шаг 3. Вывести меню.

Шаг 4. Ввести menu

Шаг 5. Очистить экран.

Шаг 6. Если menu==1 ,то

Выполнить функцю для создания меню односвязных списков.

Шаг 12.

Шаг 7. Если menu==2 ,то

Выполнить функцю для создания меню двухсвязных списков.

Шаг 12.

Шаг 8. Если menu==3 ,то

Выполнить функцю для создания меню двухсвязных кольцевых списков.

Шаг 12.

Шаг 9. Если menu==4 ,то

Вывести справку.

Шаг 12.

Шаг 10. Если menu==0 ,то

Шаг 12.

Шаг 11. Вывести сообщение об ошибке.

Шаг 12. Если menu != 0 то Шаг 2, иначе конец программы.

# 7.Описание функций

# 7.1 Функция my\_getch

**Назначение функции:**

Это функция для получения нажатой клавиши.

**Прототип:**

char my\_getch();

**Возвращаемое значение:**

Эта функция возвращает нажатую пользователем кнопку.

**Пример вызова:**

ch=my\_getch();

**Описание переменных:**

Описание переменных в таблице 8. Переменные функции my\_getch.

Таблица 8. Переменные функции my\_getch

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ch | char | Символ нажатой клавиши |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | ch | char | Символ нажатой клавиши |

# 7.2 Функция input\_number

**Назначение функции:**

Это функция для ввода целого числа.

**Прототип:**

int input\_number();

**Возвращаемое значение:**

Эта функция возвращает введённое число.

**Пример вызова:**

i=input\_number()

**Описание переменных:**

Описание переменных в таблице 9. Переменные функции str\_emepty

Таблица 9. Переменные функции str\_emepty

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | i | int | Введённое число |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | 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:**

Это функция для ввода ответа: да или нет.

**Прототип:**

bool input\_yn();

**Возвращаемое значение:**

Эта функция возвращает 0 или 1.

**Пример вызова:**

input\_yn()

**Описание переменных:**

Описание переменных в таблице 10. Переменные функции input\_yn

Таблица 10. Переменные функции input\_yn

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | i | int | Введённое число |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | sy | char | Введённый символ |
| ch | char | Нажатая клавиша |

# 7.4 Функция input\_string

**Назначение функции:**

Это функция для ввода строки.

**Прототип:**

char \*input\_string(int size);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый символ введённой строки.

**Пример вызова:**

s=input\_string(30);

**Описание переменных:**

Описание переменных в таблице 11. Переменные функции input\_string

Таблица 11. Переменные функции input\_string

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | size | int | Максимальная длинна строки |
| **Формальные параметры** | s | char \* | Введённая строка |
| **Локальные переменные** | s | char \* | Указатель на первый символ исходной строки |
| length | int | Длина строки |
| ch | char | Введённый символ |

# 7.5 Функция str\_len

**Назначение функции:**

Это функция для вычисления длинны строки.

**Прототип:**

char \*str\_len (int size);

**Возвращаемое значение:**

Эта функция возвращает длину строки

**Пример вызова:**

str\_len ("sss")

**Описание переменных:**

Описание переменных в таблице 12. Переменные функции str\_len

Таблица 12. Переменные функции str\_len

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | Фактические переменные отсутствуют | | |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | i | int | Длина строки |

# 7.6 Функция get\_type

**Назначение функции:**

Это функция для ввода типа кораблика.

**Прототип:**

char\* get\_type();

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент введённой строки.

**Пример вызова:**

s=get\_type();

**Описание переменных:**

Описание переменных в таблице 13. Переменные функции get\_type

Таблица 13. Переменные функции get\_type

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | s | char \* | Указатель на первый символ строки |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | i | int | Вспомогательная переменная |
| s | char\* | Указатель на первый символ вводимой строки |
| type | char | Тип кораблика |
| len | char | Длина строки |
| types | char\* | Массив указателей на первый символ строки |

# 7.7 Функция get\_name

**Назначение функции:**

Это функция для ввода имени кораблика.

**Прототип:**

char\* get\_name();

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент введённой строки.

**Пример вызова:**

s= get\_name();

**Описание переменных:**

Описание переменных в таблице 14. Переменные функции get\_name

Таблица 14. Переменные функции get\_name

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | s | char \* | Указатель на первый символ строки |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | s | char\* | Указатель на первый символ вводимой строки |
| exit | bool | Флажок выхода |

# 7.8 Функция get\_coord

**Назначение функции:**

Это функция для ввода координат.

**Прототип:**

int get\_coord(char ch);

**Возвращаемое значение:**

Эта функция возвращает введённое число.

**Пример вызова:**

i=get\_coord('z');

**Описание переменных:**

Описание переменных в таблице 15. Переменные функции get\_coord

Таблица 15. Переменные функции get\_coord

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | i | int | Введённое число |
| **Формальные параметры** | ch | char | Название координаты |
| **Локальные переменные** | Локальные переменные отсутствуют | | |

# 7.9 Функция get\_struct\_info

**Назначение функции:**

Это функция для получения информации о структуре.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция ничего возвращает.

**Пример вызова:**

get\_struct\_info(&info);

**Описание переменных:**

Описание переменных в таблице 16. Переменные функции get\_struct\_info

Таблица 16. Переменные функции get\_struct\_info

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | info | DATA | Структура данных кораблика |
| **Формальные параметры** | info | DATA\* | Указатель на структуру данных кораблика |
| **Локальные переменные** | Локальные переменные отсутствуют | | |

# 7.10 Функция select\_sort

**Назначение функции:**

Это функция для выбора параметров сортировки.

**Прототип:**

void select\_sort(SORTDATA \*sort);

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

select\_sort(&sort);

**Описание переменных:**

Описание переменных в таблице 17. Переменные функции select\_sort

Таблица 17. Переменные функции select\_sort

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | sort | SORTDATA | Данные о сортировке |
| **Формальные параметры** | sort | SORTDATA\* | Указатель на структуру |
| **Локальные переменные** | b | bool | Вспомогательная переменная |
| i | int | Вспомогательная переменная |
| len | int | Количество полей |
| r | int | Вспомогательная переменная |
| o | int | Вспомогательная переменная |

# 7.11 Функция select\_find

**Назначение функции:**

Это функция для выбора параметров поиска.

**Прототип:**

void select\_find(FINDDATA \*find);

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

select\_find(&find);

**Описание переменных:**

Описание переменных в таблице 18. Переменные функции select\_find

Таблица 18. Переменные функции select\_find

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | find | FINDDATA | Данные о сортировке |
| **Формальные параметры** | find | FINDDATA\* | Указатель на структуру |
| **Локальные переменные** | typ | char | Массив размера 5 данных о сортировке |
| i | int | Вспомогательная переменная |
| len | int | Количество полей, по которым будет проводиться поиск |
| r | int | Вспомогательная переменная |
| o | int | Вспомогательная переменная |

# 7.12 Функция a\_sma\_b

**Назначение функции:**

Это функция для сравнения двух элементов.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция возвращает 1 если a<b.

**Пример вызова:**

a\_sma\_b(a, b, &sort)

**Описание переменных:**

Описание переменных в таблице 19. Переменные функции a\_sma\_b

Таблица 19. Переменные функции a\_sma\_b

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | a | DATA | Данные кораблике |
| b | DATA | Данные кораблике |
| sort\_t | SORTDATA\* | Указатель на данные о сортировке |
| **Формальные параметры** | a | DATA | Данные кораблике |
| b | DATA | Данные кораблике |
| sort\_t | SORTDATA | Данные о сортировке |
| **Локальные переменные** | state | char | Значение результата(1 - a<b, 0 - a==b, -1 - a>b) |
| i | int | Вспомогательная переменная |

# 7.13 Функция is\_need

**Назначение функции:**

Это функция для определения, подходит ли элемент критериям поиска.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция возвращает 1 если подходит.

**Пример вызова:**

is\_need(info,&find)

**Описание переменных:**

Описание переменных в таблице 20. Переменные функции is\_need

Таблица 20. Переменные функции is\_need

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | info | DATA | Структура данных кораблика |
| find | FINDDATA | Структура данных о поиске |
| **Формальные параметры** | info | DATA | Структура данных кораблика |
| find | FINDDATA\* | Указатель на структуру данных о поиске |
| **Локальные переменные** | Локальные переменные отсутствуют | | |

# 7.14 Функция O\_get\_structs

**Назначение функции:**

Это функция для ввода списка.

**Прототип:**

OSHIP \*O\_get\_structs();

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=O\_get\_structs();

**Описание переменных:**

Описание переменных в таблице 21. Переменные функции O\_get\_structs

Таблица 21. Переменные функции O\_get\_structs

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | n\_sp | OSHIP \* | Указатель на первый элемент списка |
| n\_el | OSHIP \* | Указатель на элемент списка |
| last\_el | OSHIP \* | Указатель на последний элемент списка |
| is\_end\_input | bool | Вспомогательная переменная |

# 7.15 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)

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=O\_add\_frist(ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 22. Переменные функции O\_add\_frist

Таблица 22. Переменные функции O\_add\_frist

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

# 7.16 Функция O\_add\_last

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship= O\_add\_last (ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 23. Переменные функции O\_add\_last

Таблица 23. Переменные функции O\_add\_last

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

# 7.17 Функция O\_add\_after

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция для ничего не возвращает.

**Пример вызова:**

ship= O\_add\_after (ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 24. Переменные функции O\_add\_after

Таблица 24. Переменные функции O\_add\_after

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

# 7.18 Функция O\_add

**Назначение функции:**

Это функция для добавления элементов в список.

**Прототип:**

OSHIP \*O\_add(OSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=O\_add(ship);

**Описание переменных:**

Описание переменных в таблице 25. Переменные функции O\_add

Таблица 25. Переменные функции O\_add

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | menu | int | Выбранный пункт меню |
| n | int | Номер элемента |
| len | int | Количество элементов в списке |
| new\_sp | OSHIP\* | Указатель на первый элемент списка |
| link | OSHIP\* | Вспомогательный указатель |

# 7.19 Функция O\_del\_one

**Назначение функции:**

Это функция для удаления одного элемента и списка.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

O\_del\_one (&ship);

**Описание переменных:**

Описание переменных в таблице 26. Переменные функции O\_add

Таблица 26. Переменные функции O\_add

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | link | OSHIP \*\* | Указатель на адрес удаляемого элемента списка |
| **Локальные переменные** | l | OSHIP\* | Вспомогательный указатель |

# 7.20 Функция O\_del

**Назначение функции:**

Это функция для удаления элементов из списка.

**Прототип:**

OSHIP \*O\_del(OSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=O\_del(ship);

**Описание переменных:**

Описание переменных в таблице 27. Переменные функции O\_del

Таблица 27. Переменные функции O\_del

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | len | int | Количество элементов в списке |
| m | int | Номер последнего удаляемого элемента |
| n | int | Номер первого удаляемого элемента |
| i | int | Вспомогательная переменная |
| link | OSHIP\* | Вспомогательный указатель |
| link\_l | OSHIP\*\* | Вспомогательный указатель |

# 7.21 Функция O\_create\_copy

**Назначение функции:**

Это функция для создания копии элемента.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

el=O\_create\_copy(ship);

**Описание переменных:**

Описание переменных в таблице 28. Переменные функции O\_create\_copy

Таблица 28. Переменные функции O\_create\_copy

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

# 7.22 Функция O\_find

**Назначение функции:**

Это функция для поиска по задаваемым критериям.

**Прототип:**

OSHIP\* O\_find(OSHIP\* ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=O\_find(ship);

**Описание переменных:**

Описание переменных в таблице 29. Переменные функции O\_find

Таблица 29. Переменные функции O\_find

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | find | FINDDATA | Данные о поиске |
| ship\_res | OSHIP\* | Указатель на первый элемент обработанного списка |
| link | OSHIP\* | Вспомогательный указатель |
| new\_el | OSHIP\* | Указатель на копию элемента |

# 7.23 Функция O\_insert

**Назначение функции:**

Это функция для вставления элемента в отсортированный список.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

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

**Описание переменных:**

Описание переменных в таблице 30. Переменные функции O\_insert

Таблица 30. Переменные функции O\_insert

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | link | OSHIP\* | Вспомогательный указатель |
| ship | OSHIP\* | Указатель на первый элемент списка |
| sort | SORTDATA\* | Указатель на данные о сортировке |
| **Формальные параметры** | ship | OSHIP\* | Вспомогательный указатель |
| el | OSHIP\* | Указатель на вставляемый элемент списка |
| sort | SORTDATA\* | Указатель на данные о сортировке |
| **Локальные переменные** | l | OSHIP\* | Вспомогательный указатель |

# 7.24 Функция O\_sort

**Назначение функции:**

Это функция для сортировки списка по задаваемым критериям.

**Прототип:**

OSHIP \*O\_sort(OSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент отсортированного массива

**Пример вызова:**

ship=O\_sort(ship);

**Описание переменных:**

Описание переменных в таблице 31. Переменные функции O\_sort

Таблица 31. Переменные функции O\_sort

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | ship\_sorted | OSHIP\* | Указатель на первый элемент отсортированного списка |
| link | OSHIP\* | Вспомогательный указатель |
| sort | SORTDATA | Данные о сортировке |

# 7.25 Функция O\_print\_struct

**Назначение функции:**

Это функция для вывода одного элемента списка.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

O\_print\_struct(ship);

**Описание переменных:**

Описание переменных в таблице 32. Переменные функции O\_print\_struct

Таблица 32. Переменные функции O\_print\_struct

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | Локальные переменные отсутствуют | | |

# 7.26 Функция O\_print

**Назначение функции:**

Это функция для вывода всего списка.

**Прототип:**

void O\_print(OSHIP \*ship);

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

O\_print(ship);

**Описание переменных:**

Описание переменных в таблице 33. Переменные функции O\_print

Таблица 33. Переменные функции O\_print

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | i | int | Вспомогательная переменная |

# 7.27 Функция O\_free\_all

**Назначение функции:**

Это функция для освобождения памяти из под списка.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция возвращает NULL.

**Пример вызова:**

ship=O\_free\_all(ship);

**Описание переменных:**

Описание переменных в таблице 34. Переменные функции O\_free\_all

Таблица 34. Переменные функции O\_free\_all

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | OSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | link | OSHIP\* | Вспомогательный указатель |

# 7.28 Функция O\_menu

**Назначение функции:**

Это функция для создания меню.

**Прототип:**

void O\_menu();

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

O\_menu();

**Описание переменных:**

Описание переменных в таблице 35. Переменные функции O\_menu

Таблица 35. Переменные функции O\_menu

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | Фактические переменные отсутствуют. | | |
| **Формальные параметры** | Формальные переменные отсутствуют. | | |
| **Локальные переменные** | ship | OSHIP\* | Указатель на первый элемент списка исходных данных |
| ship\_res | OSHIP\* | Указатель на первый элемент списка результатов поиска |
| menu | int | Выбранный пункт меню |
| i | int | Вспомогательная переменная |
| is\_find | bool | Флажок поиска |

# 7.29 Функция D\_get\_structs

**Назначение функции:**

Это функция для ввода списка.

**Прототип:**

OSHIP \*D\_get\_structs();

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=D\_get\_structs();

**Описание переменных:**

Описание переменных в таблице 36. Переменные функции D\_get\_structs

Таблица 36. Переменные функции D\_get\_structs

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | n\_sp | DSHIP \* | Указатель на первый элемент списка |
| last\_el | DSHIP \* | Указатель на последний элемент списка |
| is\_end\_input | bool | Вспомогательная переменная |

# 7.30 Функция D\_add\_frist

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

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

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=D\_add\_frist(ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 37. Переменные функции D\_add\_frist

Таблица 37. Переменные функции D\_add\_frist

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| add\_sp | DSHIP\* | Указатель на первый элемент добавляемого списка |
| **Формальные параметры** | sp | DSHIP\* | Указатель на первый элемент списка |
| add\_sp | DSHIP\* | Указатель на первый элемент добавляемого списка |
| **Локальные переменные** | next | DSHIP\* | Вспомогательный указатель |

# 7.31 Функция D\_add\_last

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

DSHIP\* D\_add\_last (DSHIP \*sp, DSHIP \*add\_sp)

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship= D\_add\_last (ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 38. Переменные функции D\_add\_last

Таблица 38. Переменные функции D\_add\_last

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| add\_sp | DSHIP\* | Указатель на первый элемент добавляемого списка |
| **Формальные параметры** | sp | DSHIP\* | Указатель на первый элемент списка |
| add\_sp | DSHIP\* | Указатель на первый элемент добавляемого списка |
| **Локальные переменные** | next | DSHIP\* | Вспомогательный указатель |

# 7.32 Функция D\_add\_after

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

DSHIP\* D\_add\_after (DSHIP \*sp, DSHIP \*add\_sp)

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship= D\_add\_after (ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 39. Переменные функции D\_add\_after

Таблица 39. Переменные функции D\_add\_after

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| add\_sp | DSHIP\* | Указатель на первый элемент добавляемого списка |
| **Формальные параметры** | sp | DSHIP\* | Указатель на первый элемент списка |
| add\_sp | DSHIP\* | Указатель на первый элемент добавляемого списка |
| **Локальные переменные** | next | DSHIP\* | Вспомогательный указатель |

# 7.33 Функция D\_add

**Назначение функции:**

Это функция для добавления элементов в список.

**Прототип:**

DSHIP \*D\_add(DSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=D\_add(ship);

**Описание переменных:**

Описание переменных в таблице 40. Переменные функции D\_add

Таблица 40. Переменные функции D\_add

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | menu | int | Выбранный пункт меню |
| n | int | Вспомогательная переменная |
| len | int | Количество элементов в списке |
| new\_sp | DSHIP\* | Указатель на первый элемент списка |
| link | DSHIP\* | Вспомогательный указатель |

# 7.34 Функция D\_del\_one

**Назначение функции:**

Это функция для удаления одного элемента из списка.

**Прототип:**

void D\_del\_one(DSHIP \*\*link)

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

D\_del\_one (&ship);

**Описание переменных:**

Описание переменных в таблице 41. Переменные функции D\_add

Таблица 41. Переменные функции D\_add

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | link | DSHIP \*\* | Указатель на адрес первого элемента списка |
| **Локальные переменные** | l | DSHIP\* | Вспомогательный указатель |

# 7.35 Функция D\_del

**Назначение функции:**

Это функция для удаления элементов из списка.

**Прототип:**

DSHIP \*D\_del(DSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=D\_del(ship);

**Описание переменных:**

Описание переменных в таблице 42. Переменные функции D\_del

Таблица 42. Переменные функции D\_del

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | len | int | Количество элементов в списке |
| m | int | Номер последнего удаляемого элемента |
| n | int | Номер первого удаляемого элемента |
| i | int | Вспомогательная переменная |
| link | DSHIP\* | Вспомогательный указатель |
| link\_l | DSHIP\*\* | Вспомогательный указатель |

# 7.36 Функция D\_create\_copy

**Назначение функции:**

Это функция для создания копии элемента.

**Прототип:**

DSHIP \*D\_create\_copy(DSHIP\* ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

el=D\_create\_copy(ship);

**Описание переменных:**

Описание переменных в таблице 43. Переменные функции D\_create\_copy

Таблица 43. Переменные функции D\_create\_copy

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

# 7.37 Функция D\_find

**Назначение функции:**

Это функция для поиска по задаваемым критериям.

**Прототип:**

DSHIP\* D\_find(DSHIP\* ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=D\_find(ship);

**Описание переменных:**

Описание переменных в таблице 44. Переменные функции D\_find

Таблица 44. Переменные функции D\_find

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | find | FINDDATA | Данные о поиске |
| ship\_res | DSHIP\* | Указатель на первый элемент обработанного списка |
| link | DSHIP\* | Вспомогательный указатель |
| new\_el | DSHIP\* | Указатель на копию элемента |

# 7.38 Функция D\_insert

**Назначение функции:**

Это функция для вставления элемента в отсортированный список.

**Прототип:**

DSHIP \*D\_insert(DSHIP \*ship, DSHIP \*el, SORTDATA\*sort);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=D\_insert(ship,link,sort);

**Описание переменных:**

Описание переменных в таблице 45. Переменные функции D\_insert

Таблица 45. Переменные функции D\_insert

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | link | DSHIP\* | Вспомогательный указатель |
| ship | DSHIP\* | Указатель на первый элемент списка |
| sort | SORTDATA\* | Указатель на данные о сортировке |
| **Формальные параметры** | ship | DSHIP\* | Вспомогательный указатель |
| el | DSHIP\* | Указатель на вставляемый элемент списка |
| sort | SORTDATA\* | Указатель на данные о сортировке |
| **Локальные переменные** | l | DSHIP\* | Вспомогательный указатель |

# 7.39 Функция D\_sort

**Назначение функции:**

Это функция для сортировки списка по задаваемым критериям.

**Прототип:**

DSHIP \*D\_sort(DSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент отсортированного списка.

**Пример вызова:**

ship=D\_sort(ship);

**Описание переменных:**

Описание переменных в таблице 46. Переменные функции D\_sort

Таблица 46. Переменные функции D\_sort

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | ship\_sorted | OSHIP\* | Указатель на первый элемент отсортированного списка |
| link | OSHIP\* | Вспомогательный указатель |
| sort | SORTDATA | Данные о сортировке |

# 7.40 Функция D\_print\_struct

**Назначение функции:**

Это функция для вывода одного элемента списка.

**Прототип:**

void D\_print\_struct(DSHIP \*ship);

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

D\_print\_struct(ship);

**Описание переменных:**

Описание переменных в таблице 47. Переменные функции D\_print\_struct

Таблица 47. Переменные функции D\_print\_struct

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | Локальные переменные отсутствуют | | |

# 7.41 Функция D\_print

**Назначение функции:**

Это функция для вывода всего списка.

**Прототип:**

void D\_print(DSHIP \*ship);

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

D\_print(ship);

**Описание переменных:**

Описание переменных в таблице 48. Переменные функции D\_print

Таблица 48. Переменные функции D\_print

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | i | int | Вспомогательная переменная |
| r | int | Вспомогательная переменная |
| type | int | Тип вывода |

# 7.42 Функция D\_free\_all

**Назначение функции:**

Это функция для освобождения памяти из под списка.

**Прототип:**

OSHIP\* D\_free\_all(DSHIP\* ship);

**Возвращаемое значение:**

Эта функция возвращает NULL.

**Пример вызова:**

ship=D\_free\_all(ship);

**Описание переменных:**

Описание переменных в таблице 49. Переменные функции D\_free\_all

Таблица 49. Переменные функции D\_free\_all

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | DSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | link | DSHIP\* | Вспомогательный указатель |

# 7.43 Функция D\_menu

**Назначение функции:**

Это функция для создания меню.

**Прототип:**

void D\_menu();

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

D\_menu();

**Описание переменных:**

Описание переменных в таблице 50. Переменные функции D\_menu

Таблица 50. Переменные функции D\_menu

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | Фактические переменные отсутствуют. | | |
| **Формальные параметры** | Формальные переменные отсутствуют. | | |
| **Локальные переменные** | ship | DSHIP\* | Указатель на первый элемент списка исходных данных |
| ship\_res | DSHIP\* | Указатель на первый элемент списка результатов поиска |
| menu | int | Выбранный пункт меню |
| i | int | Вспомогательная переменная |
| is\_find | bool | Флажок поиска |

# 7.44 Функция C\_add\_frist

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

CSHIP\* S\_add\_frist(CSHIP \*sp, CSHIP \*add\_sp)

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=C\_add\_frist(ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 51. Переменные функции C\_add\_frist

Таблица 51. Переменные функции C\_add\_frist

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| add\_sp | CSHIP\* | Указатель на первый элемент добавляемого списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| sp | CSHIP\* | Указатель на первый элемент добавляемого списка |
| **Локальные переменные** | l | CSHIP\* | Вспомогательный указатель |

# 7.45 Функция C\_add\_last

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

CSHIP\* C\_add\_last (CSHIP \*sp, CSHIP \*add\_sp)

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship= C\_add\_last (ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 52. Переменные функции C\_add\_last

Таблица 52. Переменные функции C\_add\_last

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| add\_sp | CSHIP\* | Указатель на первый элемент добавляемого списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| sp | CSHIP\* | Указатель на первый элемент добавляемого списка |
| **Локальные переменные** | l | CSHIP\* | Вспомогательный указатель |

# 7.46 Функция C\_add\_after

**Назначение функции:**

Это функция для добавления списка в начало исходного списка.

**Прототип:**

CSHIP\* C\_add\_after (CSHIP \*sp, CSHIP \*add\_sp)

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship= C\_add\_after (ship,add\_sp);

**Описание переменных:**

Описание переменных в таблице 53. Переменные функции C\_add\_after

Таблица 53. Переменные функции C\_add\_after

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| add\_sp | DSHIP\* | Указатель на первый элемент добавляемого списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| sp | CSHIP\* | Указатель на первый элемент добавляемого списка |
| **Локальные переменные** | l | CSHIP\* | Вспомогательный указатель |

# 7.47 Функция C\_get\_structs

**Назначение функции:**

Это функция для ввода списка.

**Прототип:**

CSHIP \*C\_get\_structs();

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=C\_get\_structs();

**Описание переменных:**

Описание переменных в таблице 54. Переменные функции C\_get\_structs

Таблица 54. Переменные функции C\_get\_structs

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | n\_sp | CSHIP \* | Указатель на первый элемент списка |
| last\_el | CSHIP \* | Указатель на последний элемент списка |
| new\_el | CSHIP \* | Указатель на введённую структуру |
|  | is\_end\_input | bool | Вспомогательная переменная |

# 7.48 Функция C\_get\_len

**Назначение функции:**

Это функция для вычисления длинны списка.

**Прототип:**

int C\_get\_len(CSHIP\*ship);

**Возвращаемое значение:**

Эта функция возвращает длинну списка.

**Пример вызова:**

len=C\_get\_len(ship);

**Описание переменных:**

Описание переменных в таблице 55. Переменные функции C\_get\_len

Таблица 55. Переменные функции C\_get\_len

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| len | int | Количество элементов |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | i | int | Вспомогательная переменная |
| link | OSHIP\* | Вспомогательный указатель |

# 7.49 Функция C\_add

**Назначение функции:**

Это функция для добавления элементов в список.

**Прототип:**

CSHIP \*C\_add(CSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=C\_add(ship);

**Описание переменных:**

Описание переменных в таблице 56. Переменные функции C\_add

Таблица 56. Переменные функции C\_add

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | menu | int | Выбранный пункт меню |
| n | int | Вспомогательная переменная |
| len | int | Количество элементов в списке |
| new\_sp | CSHIP\* | Указатель на первый элемент списка |
| link | CSHIP\* | Вспомогательный указатель |

# 7.50 Функция C\_del\_one

**Назначение функции:**

Это функция для удаления одного элемента и списка.

**Прототип:**

CSHIP\* C\_del\_one(CSHIP \*link);

**Возвращаемое значение:**

Эта функция возвращает указатель на следующий элемент.

**Пример вызова:**

C\_del\_one (ship);

**Описание переменных:**

Описание переменных в таблице 57. Переменные функции C\_del\_one

Таблица 57. Переменные функции C\_del\_one

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | link | CSHIP \* | Указатель на адрес первого элемента списка |
| **Локальные переменные** | Локальные переменные отсутствуют | | |

# 7.51 Функция C\_del

**Назначение функции:**

Это функция для удаления элементов из списка.

**Прототип:**

CSHIP \*C\_del(CSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=C\_del(ship);

**Описание переменных:**

Описание переменных в таблице 58. Переменные функции C\_del

Таблица 58. Переменные функции C\_del

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | len | int | Количество элементов в списке |
| m | int | Номер последнего удаляемого элемента |
| n | int | Номер первого удаляемого элемента |
| i | int | Вспомогательная переменная |
| link | CSHIP\* | Вспомогательный указатель |

# 7.52 Функция C\_print\_struct

**Назначение функции:**

Это функция для вывода элемента списка.

**Прототип:**

void C\_print\_struct(CSHIP \*ship);

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

C\_print\_struct(ship);

**Описание переменных:**

Описание переменных в таблице 59. Переменные функции C\_print\_struct

Таблица 59. Переменные функции C\_print\_struct

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на выводимый элемент списка |
| **Локальные переменные** | Локальные переменные отсутствуют | | |

# 7.53 Функция C\_print

**Назначение функции:**

Это функция для вывода всего списка.

**Прототип:**

void C\_print(CSHIP \*ship);

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

C\_print(ship);

**Описание переменных:**

Описание переменных в таблице 60. Переменные функции C\_print

Таблица 60. Переменные функции C\_print

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | i | int | Вспомогательная переменная |
| r | int | Вспомогательная переменная |
| a | int | Вспомогательная переменная |
| type | int | Тип вывода(1-в одну сторону 2-в другую) |
| link | CSHIP\* | Вспомогательный указатель |
| len | int | Количество элементов в списке |
| n | int | Номер элемента, с которого начинается вывод |
| linkadd | int | Количество битов, которое надо прибавить к указателю на элемент списка, чтобы получить адрес следующего выводимого элемента |

# 7.54 Функция C\_free\_all

**Назначение функции:**

Это функция для освобождения памяти из под списка.

**Прототип:**

OSHIP\* C\_free\_all(CSHIP\* ship);

**Возвращаемое значение:**

Эта функция возвращает NULL.

**Пример вызова:**

ship=C\_free\_all(ship);

**Описание переменных:**

Описание переменных в таблице 61. Переменные функции C\_free\_all

Таблица 61. Переменные функции C\_free\_all

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | link | CSHIP\* | Вспомогательный указатель |

# 7.55 Функция C\_create\_copy

**Назначение функции:**

Это функция для создания копии элемента.

**Прототип:**

CSHIP \*C\_create\_copy(CSHIP\* ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на копированный элемент списка.

**Пример вызова:**

el=C\_create\_copy(ship);

**Описание переменных:**

Описание переменных в таблице 62. Переменные функции C\_create\_copy

Таблица 62. Переменные функции C\_create\_copy

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на элемент списка |
| el | CSHIP\* | Указатель на элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | new\_ship | CSHIP\* | Указатель на копированный элемент |
| i | int | Вспомогательная переменная |
| len | int | Количество элементов в списке |

# 7.56 Функция C\_insert

**Назначение функции:**

Это функция для вставления элемента в отсортированный список.

**Прототип:**

CSHIP \*C\_insert(CSHIP \*ship, CSHIP \*el, SORTDATA\*sort);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship\_sorted = C\_insert(ship\_sorted, link, &sort);

**Описание переменных:**

Описание переменных в таблице 63. Переменные функции C\_insert

Таблица 63. Переменные функции C\_insert

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship\_sorted | CSHIP\* | Указатель на первый элемент отсортированного списка |
| link | CSHIP\* | Вспомогательный указатель |
| sort | SORTDAT | Данные о сортировке |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| el | CSHIP \* | Вставляемый элемент. |
| sort | SORTDATA\* | Указатель на данные о сортировке |
| **Локальные переменные** | l | CSHIP \* | Вспомогательный указатель |

# 7.57 Функция C\_sort

**Назначение функции:**

Это функция для сортировки списка по задаваемым критериям.

**Прототип:**

CSHIP \*C\_sort(CSHIP \*ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент отсортированного списка.

**Пример вызова:**

ship=C\_sort(ship);

**Описание переменных:**

Описание переменных в таблице 64. Переменные функции C\_sort

Таблица 64. Переменные функции C\_sort

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | link | CSHIP\* | Вспомогательный указатель |
| sort | SORTDATA | Данные о сортировке |
| ship\_sorted | CSHIP\* | Указатель на первый элемент отсортированного списка |

# 7.58 Функция C\_find

**Назначение функции:**

Это функция для поиска по задаваемым критериям.

**Прототип:**

CSHIP\* C\_find(CSHIP\* ship);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый элемент списка.

**Пример вызова:**

ship=C\_find(ship);

**Описание переменных:**

Описание переменных в таблице 65. Переменные функции C\_find

Таблица 65. Переменные функции C\_find

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Формальные параметры** | ship | CSHIP\* | Указатель на первый элемент списка |
| **Локальные переменные** | find | FINDDATA | Данные о поиске |
| ship\_res | CSHIP\* | Указатель на первый элемент обработанного списка |
| link | CSHIP\* | Вспомогательный указатель |
| new\_el | CSHIP\* | Вспомогательный указатель |

# 7.59 Функция C\_menu

**Назначение функции:**

Это функция для создания меню.

**Прототип:**

void C\_menu();

**Возвращаемое значение:**

Эта функция ничего не возвращает.

**Пример вызова:**

C\_menu();

**Описание переменных:**

Описание переменных в таблице 66. Переменные функции C\_menu

Таблица 66. Переменные функции C\_menu

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | Фактические переменные отсутствуют. | | |
| **Формальные параметры** | Формальные переменные отсутствуют. | | |
| **Локальные переменные** | ship | CSHIP\* | Указатель на первый элемент списка исходных данных |
| ship\_res | CSHIP\* | Указатель на первый элемент списка результатов поиска |
| menu | int | Выбранный пункт меню |
| i | int | Вспомогательная переменная |
| is\_find | bool | Флажок поиска |

# 8.Текст программы

#include <conio.h>

#include <malloc.h>

#include <stdio.h>

#include <windows.h>

#include <locale.h>

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 typ[5];//Массив размера 5, в котором элемент означает критерий сортировки (1-имя

2-тип 3-x 4-y 5-z 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

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)

} FINDDATA;

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

}DATA;

//Функция для получения символа.

char my\_getch()

{

char ch;//Код нажатой кнопки.

do

{

ch = \_getch();

if (ch == '\r')

ch = '\n';

else

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

\_getch();

else

if (ch == '\t')

ch = ' ';

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

return ch;

}

//Функция для ввода целого числа

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 = NULL;

do

{

ch = my\_getch();

switch (ch)

{

case 27:

break;

case '\b':

if (length > 1)

{

printf("\b \b");

length--;

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

}

break;

case '\n':

if (length>1)

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;

}

//Функция для ввода ответа y/n

bool input\_yn()

{

char sy; //Введённый символ

char ch; //Нажатая клавиша

sy = 0;

do

{

ch = my\_getch();

switch (ch)

{

case '\b':

printf("\b \b");

sy = 0;

break;

case 'y':

case 'Y':

case 'n':

case 'N':

sy = ch;

printf("\b%c",sy);

break;

}

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

printf("\n");

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

}

//Функция для ввода строки

char \*input\_string(int size)

{

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

int length = 1;//Длина строки

char ch; //Введённый символ

s = NULL;

do

{

ch = my\_getch();

switch (ch)

{

case 27:

break;

case '\b':

if (length > 1)

{

printf("\b \b");

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 (length <= size && (ch != ' ' || length > 1))

{

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;

}

//Функция для вычисления длинны строки

int str\_len(char\*s)

{

int i = 1;//Длина строки

while (\*s)

i++, s++;

return i;

}

//Функция для ввода типа кораблика

char\* get\_type()

{

char \*types[4] = { "battleship", "cruiser", "boat", "steelship" };

//Массив указателей на первый символ строки

int i; //Вспомогательная переменная

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

char type;//Тип ораблика

char len; //Длина строки

puts("Выберите тип:");

for (i = 0; i < 4; i++)

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

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

type = input\_number();

if (type <= 0 || type >= 5)

{

puts("Вводите тип (от 1 до 20 символов):");

s = input\_string(20);

}

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;

}

//Функция для ввода имени

char\* get\_name()

{

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

bool exit;//Флажок выхода

exit = 0;

do{

puts("Введите имя (имя должно содержать хотя бы один символ");

puts("и не более 30 символов):");

s = input\_string(30);

if (\*s < 'A' || \*s > 'Z')

{

puts("Вниание! Имя корабля начинается не с большой буквы.");

puts("Вы хотите ввести другое имя? (y/n)");

if (input\_yn())

free(s);

else

exit = 1;

}

else

exit = 1;

} while (!exit);

return s;

}

//Функция для ввода координат

int get\_coord(char ch)

{

printf("Вводите положение по %c в сантиметрах (целое число от -999999 до 9999999):\n",

ch);

return input\_number();

}

//Функция получения полей структуры

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

{

info->name = get\_name();

info->type = get\_type();

info->x = get\_coord('x');

info->y = get\_coord('y');

info->z = get\_coord('z');

}

//Функция для выбора типа сортировки

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 5)?");

len = input\_number();

if (len < 1 || len > 5)

puts("Ошибка");

} while (len < 1 || len > 5);

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");

r = input\_number();

if (r < 1 || r > 5)

puts("Ошибка");

} while (r< 1 || r > 5);

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 > 5)

puts("Ошибка");

} while (r< 1 || r > 5);

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;

}

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");

r = input\_number();

if (r < 1 || r > 5)

puts("Нет такого пункта меню");

} while (r < 1 || r > 5);

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 > 5)

puts("Ошибка");

} while (r< 1 || r > 5);

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;

}

}

if (len < 5)

sort->typ[len] = 0;

}

//Функция для выбора типа поиска

void select\_find(FINDDATA \*find)

{

int i; //Вспомогательная переменная

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

int r; //Вспомогательная переменная

char typ[5];//Массив размера 5 данных о сортировке

int o; //Вспомогательная переменная

for (i = 0; i < 5; 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 5)?");

len = input\_number();

if (len < 1 || len > 5)

puts("Ошибка");

} while (len < 1 || len > 5);

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");

r = input\_number();

if (r < 1 || r > 5)

puts("Ошибка");

} while (r < 1 || r > 5);

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\_name();

break;

case 2:

find->type = get\_type();

break;

case 3:

find->x = get\_coord('x');

break;

case 4:

find->y = get\_coord('y');

break;

case 5:

find->z = get\_coord('z');

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;

}

i++;

} while (i<len);

}

//Функция для сравнения двух элементов

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 < 5 && 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;

}

}

return state == 1;

}

//Функция для определения, подходит ли элемент критериям поиска

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;

}

return 1;

}

//@@

typedef struct ST1

{

DATA info; //Данные о кораблике

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

} OSHIP;

//Функция для ввода списка

OSHIP \*O\_get\_structs()

{

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

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

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

bool is\_end\_input;//Вспомогательная переменная

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

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

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

if (input\_yn())

{

is\_end\_input = false;

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;

}

//Функция для добавления списка в начало исходного 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;

}

//Функция для добавления списка в конец исходного списка

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

{

if (sp != NULL)

{

OSHIP \*li;//Вспомогательный указатель

for (li = sp; li->link1 != NULL; li = li->link1);

li->link1 = add\_sp;

}

else

sp = add\_sp;

return sp;

}

//Функция для добавления списка после определённого элемента исходного списка

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;

}

}

//Функция для добавления новых элементов в список

OSHIP \*O\_add(OSHIP \*ship)

{

int menu; //Выбранный пункт меню

int n; //Номер элемента

int len; //Количество элементов в списке

OSHIP \*new\_sp;//Указатель на первый элемент списка

OSHIP \*link; //Вспомогательный указатель

puts("Вводите последовательность элементов");

if (ship != NULL)

{

new\_sp = O\_get\_structs();

do

{

system("cls");

puts("Вставка");

puts("1-Вставить в начало");

puts("2-Вставить в конец");

puts("3-Вставить после n-ного элемента");

menu = input\_number();

if (menu<1 || menu>4)

{

puts("Нет такого пункта");

system("pause");

}

} while (menu<1 || menu>4);

switch (menu)

{

case 1:

ship = O\_add\_frist(ship, new\_sp);

break;

case 2:

ship = O\_add\_last(ship, new\_sp);

break;

case 3:

for (len = 1, link = ship; link->link1 != NULL; link = link->link1, len++);

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");

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

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

O\_add\_after(link, new\_sp);

break;

}

}

else

ship = O\_get\_structs();

puts("Новые элементы успешно добавлены");

return ship;

}

//Функция для удаления одного элемента из списка

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

{

if (\*link != NULL)

{

OSHIP \*l;//

l = \*link;

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

free(l->info.name);

free(l->info.type);

free(l);

}

}

//Функция для удаления элементов из списка

OSHIP \*O\_del(OSHIP \*ship)

{

int len; //Номер первого удаляемгог элемента

int m; //Номер последнего удаляемого элемента

int n; //Номер первого удаляемого элемента

int i; //Вспомогательная переменная

OSHIP \*link; //Вспомогательный указатель

OSHIP \*\*link\_l;//Вспомогательный указатель

if (ship != NULL)

{

for (link = ship, len = 1; link->link1 != NULL; link = link->link1, len++);

puts("Удаление элементов с n до m");

do

{

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

n = input\_number();

if (n<1 || n>len)

puts("Неправильный ввод");

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

do

{

printf("Вводите m(от %i до %i)\n", n,len);

m = input\_number();

if (m<n || m>len)

puts("Неправильный ввод");

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

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("Элемент(-ы) успешно удалены");

}

else

puts("Список пуст, удалять нечего");

return ship;

}

//Функция для создания копии элемента списка

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;

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;

}

//Функция для поиска

OSHIP\* O\_find(OSHIP\* ship)

{

FINDDATA find; //Указатель на копию элемента

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

OSHIP \*link; //Вспомогательный указатель

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

select\_find(&find);

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;

if (ship\_res != NULL)

puts("Поиск завершон");

else

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

return ship\_res;

}

//Функция для вставления элемента в отсортированный список.

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;

}

}

//Функция для сортировки

OSHIP \*O\_sort(OSHIP \*ship)

{

if (ship == NULL)

{

puts("Нечего сортировать");

return NULL;

}

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

OSHIP \*link; //Вспомогательный указатель

SORTDATA sort; //Данные о сортировке

select\_sort(&sort);

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");

return ship\_sorted;

}

//Функция для вывода информационных полей одной структуры

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 по x в сантиметрах:%i\n", ship->info.x);

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

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

}

//Функция для вывода списка

void O\_print(OSHIP \*ship)

{

int i;//Вспомогательная переменная

puts("Вывод с первого элемента");

if (ship != NULL)

{

i = 0;

do

{

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

system("pause");

printf("\nКораблик номер: %i\n", i + 1);

O\_print\_struct(ship);

ship = ship->link1;

i++;

}

while (ship != NULL);

printf("\nВывод окончен\n");

}

else

puts("Список пуст");

}

//Функция для освобождения памяти из под списка

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

{

OSHIP \*link;//

while(ship != NULL)

{

link = ship;

ship = ship->link1;

free(link->info.name);

free(link->info.type);

free(link);

}

return NULL;

}

//Функция для создания меню

void O\_menu()

{

int menu; //Выбранный пункт меню

int i; //Вспомогательная переменная

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

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

bool is\_find; //Флажок поиска

ship = NULL;

ship\_res = NULL;

is\_find = false;

do

{

system("cls");

puts("Односвязные списки");

puts("1-добавление элементов в список");

puts("2-удаление элементов из списка");

puts("3-вывод элементов списка");

puts("4-обработка списка");

puts("5-вывод результатов последнего поиска");

puts("6-справка");

puts("0-возврат в надменю");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

ship = O\_add(ship);

system("pause");

break;

case 2:

ship = O\_del(ship);

system("pause");

break;

case 3:

O\_print(ship);

system("pause");

break;

case 4:

if (ship != NULL)

{

do

{

system("cls");

puts("Выберите тип обработки");

puts("1-поиск (при этом формируется новый список)");

puts("2-сортировка элементов исходного списка");

puts("0-возврат в надменю");

i = input\_number();

if (i<0 || i>2)

{

puts("Вы неправильно ввели пункт меню");

system("pause");

}

} while (i<0 || i>2);

switch (i)

{

case 1:

is\_find = TRUE;

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

ship\_res = O\_find(ship);

system("pause");

break;

case 2:

ship = O\_sort(ship);

system("pause");

break;

}

}

else

{

puts("Список пуст, обрабатывать нечего");

system("pause");

}

break;

case 5:

if (is\_find)

if (ship\_res != NULL)

O\_print(ship\_res);

else

puts("При последнем поиске не найдено элементов,\nсоответствующих заданным

критериям");

else

puts("Вы ещё ни разу не проводили поиск");

system("pause");

break;

case 6:

puts("Справка");

puts("При вводе строки:");

puts("1. таб конвертируется в пробел");

puts("2. пробелы сначала вводить запрещено");

puts("3. пробелы в конце убираются");

puts("Кораблик имеет пять полей:");

puts("1. Имя-строка от 1 до 30 символов");

puts("2. Тип-строка от 1 до 20 символов");

puts("3. Положение по x-целое число от -999999 до 9999999");

puts("4. Положение по y-целое число от -999999 до 9999999");

puts("5. Положение по z-целое число от -999999 до 9999999");

puts("Результат обработки - новый массив данных состоящий из элементов исходного");

puts("массива, которые соответствуют заданным критериям.");

puts("Русские буквы вводить запрещено.");

puts("Для двухсвязных списков возможен вывод с начала");

puts("и с конца");

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu != 0);

ship=O\_free\_all(ship);

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

}

//@@

typedef struct ST2

{

DATA info; //Данные о кораблике

struct ST2 \*link1;//Указатель на следующий элемент

struct ST2 \*link2;//Указатель на предыдующий элемент

} DSHIP;

DSHIP \*D\_get\_structs()

{

DSHIP\* n\_sp; //Указатель на первый элемент списка

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

bool is\_end\_input;//Вспомогательная переменная

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

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

n\_sp->link2 = NULL;

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

if (input\_yn())

{

is\_end\_input = false;

last\_el = n\_sp;

do

{

last\_el->link1 = (DSHIP\*)malloc(sizeof(DSHIP));

get\_struct\_info(&(last\_el->link1->info));

last\_el->link1->link2 = last\_el;

last\_el = last\_el->link1;

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;

}

//Функция для добавления списка в начало исходного списка

DSHIP \*D\_add\_frist(DSHIP \*ship, DSHIP \*sp)

{

if (ship != NULL)

{

DSHIP\* next;//Вспомогательный указатель

for (next = sp; next->link1 != NULL; next = next->link1);

next->link1 = ship;

ship->link2 = next;

}

return sp;

}

//Функция для добавления списка в начало исходного списка.

DSHIP \*D\_add\_last(DSHIP \*ship, DSHIP \*sp)

{

if (ship != NULL)

{

DSHIP\* next;

for (next = ship; next->link1 != NULL; next = next->link1);

next->link1 = sp;

sp->link2 = next;

}

else

ship = sp;

return ship;

}

//Функция для добавления списка в начало исходного списка.

DSHIP \*D\_add\_after(DSHIP \*ship, DSHIP \*sp)

{

if (ship != NULL)

{

DSHIP\* next;

for (next = sp; next->link1 != NULL; next = next->link1);

next->link1 = ship->link1;

if (next->link1 != NULL)

next->link1->link2 = next;

ship->link1 = sp;

sp->link2 = ship;

}

else

ship = sp;

return ship;

}

//Функция для добавления списка в начало исходного списка

DSHIP \*D\_add(DSHIP \*ship)

{

int menu; //Выбранный пункт меню

int n; //Вспомогательная переменная

int len; //Количество элементов в списке

DSHIP \*new\_sp;//Указатель на первый элемент списка

DSHIP \*link; //Указатель на первый элемент списка

puts("Вводите последовательность элементов");

if (ship != NULL)

{

new\_sp = D\_get\_structs();

do

{

system("cls");

puts("Вставка");

puts("1-Вставить в начало");

puts("2-Вставить в конец");

puts("3-Вставить после n-ного элемента");

menu = input\_number();

if (menu<1 || menu>4)

{

puts("Нет такого пункта");

system("pause");

}

} while (menu<1 || menu>4);

switch (menu)

{

case 1:

ship = D\_add\_frist(ship, new\_sp);

break;

case 2:

ship = D\_add\_last(ship, new\_sp);

break;

case 3:

for (len = 1, link = ship; link->link1 != NULL; link = link->link1, len++);

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");

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

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

D\_add\_after(link, new\_sp);

break;

}

}

else

ship = D\_get\_structs();

puts("Новые элементы успешно добавлены");

return ship;

}

//Функция для удаления одного элемента из списка

void D\_del\_one(DSHIP \*\*link)

{

if (\*link != NULL)

{

DSHIP \*l;//Вспомогательный указатель

l = \*link;

if ((\*link)->link1 != NULL)

(\*link)->link1->link2 = (\*link)->link2;

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

free(l->info.name);

free(l->info.type);

free(l);

}

}

//Функция для удаления элементов из списка

DSHIP \*D\_del(DSHIP \*ship)

{

int len; //Количество элементов в списке

int m; //Номер последнего удаляемого элемента

int n; //Номер первого удаляемого элемента

int i; //Вспомогательная переменная

DSHIP \*link; //Вспомогательный указатель

DSHIP \*\*link\_l;//Вспомогательный указатель

if (ship != NULL)

{

for (link = ship, len = 1; link->link1 != NULL; link = link->link1, len++);

puts("Удаление элементов с n до m");

do

{

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

n = input\_number();

if (n<1 || n>len)

puts("Неправильный ввод");

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

do

{

printf("Вводите m(от %i до %i)\n", n, len);

m = input\_number();

if (m<n || m>len)

puts("Неправильный ввод");

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

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

{

D\_del\_one(link\_l);

i--;

} while (i > 0);

puts("Элемент(-ы) успешно удалены");

}

else

puts("Список пуст, удалять нечего");

return ship;

}

//Функция для создания копии элемента списка

DSHIP \*D\_create\_copy(DSHIP\* ship)

{

int i; //Вспомогательная переменная

int len; //Количество элементов в списке

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

if (ship != NULL)

{

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

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

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

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

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;

new\_ship->link2 = NULL;

}

else

new\_ship = NULL;

return new\_ship;

}

//Функция для поиска

DSHIP\* D\_find(DSHIP\* ship)

{

FINDDATA find; //Данные о поиске

DSHIP \*ship\_res;//Указатель на первый элемент обработанного списка

DSHIP \*link; //Вспомогательный указатель

DSHIP \*new\_el; //Указатель на копию элемента

select\_find(&find);

ship\_res = NULL;

link = NULL;

do

{

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

{

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

if (ship\_res == NULL)

ship\_res = new\_el;

else

link->link1 = new\_el;

new\_el->link2 = link;

link = new\_el;

}

ship = ship->link1;

} while (ship != NULL);

if (ship\_res != NULL)

link->link1 = NULL;

if (ship\_res != NULL)

puts("Поиск завершон");

else

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

return ship\_res;

}

//Функция для вставки элемента в отсортированный список

DSHIP \*D\_insert(DSHIP \*ship, DSHIP \*el, SORTDATA\*sort)

{

if (ship != NULL)

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

{

DSHIP \*l;

l = ship;//Вспомогательный указатель

while (ship->link1 != NULL)

{

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

{

D\_add\_after(ship, el);

return l;

}

ship = ship->link1;

}

ship->link1 = el;

ship->link1->link2 = ship;

return l;

}

else

{

el->link1 = ship;

el->link1->link2 = el;

return el;

}

else

{

el->link1 = NULL;

el->link2 = NULL;

return el;

}

}

//Функция для сортировки

DSHIP \*D\_sort(DSHIP \*ship)

{

if (ship == NULL)

{

puts("Нечего сортировать");

return NULL;

}

DSHIP \*ship\_sorted;//

DSHIP \*link; //

SORTDATA sort; //

select\_sort(&sort);

ship\_sorted = NULL;

do

{

link = ship;

ship = ship->link1;

link->link1 = NULL;

link->link2 = NULL;

ship\_sorted = D\_insert(ship\_sorted, link, &sort);

} while (ship != NULL);

puts("Сортировка окончена");

return ship\_sorted;

}

//Функция для вывода информационных полей одной структуры

void D\_print\_struct(DSHIP \*ship)

{

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

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

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

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

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

}

//Функция для вывода списка

void D\_print(DSHIP \*ship)

{

int i; //Вспомогательная переменная

int r; //Вспомогательная переменная

int type;//Тип вывода

if (ship != NULL)

{

do

{

puts("1-Вывод с первого элемента");

puts("2-Вывод с последнего элемента");

type = input\_number();

if (type < 1 || type > 2)

puts("Нет такого пункта");

} while (type < 1 || type > 2);

if (type == 1)

{

i = 0;

do

{

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

system("pause");

i++;

printf("\nКораблик номер: %i\n", i);

D\_print\_struct(ship);

ship = ship->link1;

} while (ship != NULL);

}

else

{

i = 1;

for (; ship->link1!=NULL; ship = ship->link1,i++);

r = i % 3;

do

{

if (ship->link1!=NULL && i % 3 == r)

system("pause");

printf("\nКораблик номер: %i\n", i);

D\_print\_struct(ship);

ship = ship->link2;

i--;

} while (ship != NULL);

}

printf("\nВывод окончен\n");

}

else

puts("Список пуст");

}

//Функция для освобождения памяти из под списка

DSHIP\* D\_free\_all(DSHIP\* ship)

{

DSHIP \*link;//Вспомогательный указатель

while (ship != NULL)

{

link = ship;

ship = ship->link1;

free(link->info.name);

free(link->info.type);

free(link);

}

return NULL;

}

//Функция для создания меню

void D\_menu()

{

DSHIP \*ship; //Указатель на первый элемент списка исходных данных

DSHIP \*ship\_res;//Указатель на первый элемент списка результатов поиска

int menu; //Выбранный пункт меню

int i; //Вспомогательная переменная

bool is\_find; //Флажок поиска

is\_find = false;

ship = NULL;

ship\_res = NULL;

do

{

system("cls");

puts("Двухсвязные списки");

puts("1-добавление элементов в список");

puts("2-удаление элементов из списка");

puts("3-вывод элементов списка");

puts("4-обработка списка");

puts("5-вывод результатов последнего поиска");

puts("6-справка");

puts("0-возврат в надменю");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

ship=D\_add(ship);

system("pause");

break;

case 2:

ship = D\_del(ship);

system("pause");

break;

case 3:

D\_print(ship);

system("pause");

break;

case 4:

if (ship != NULL)

{

do

{

system("cls");

puts("Выберите тип обработки");

puts("1-поиск (при этом формируется новый список)");

puts("2-сортировка элементов исходного списка");

puts("0-возврат в надменю");

i = input\_number();

if (i<0 || i>2)

{

puts("Вы неправильно ввели пункт меню");

system("pause");

}

} while (i<0 || i>2);

switch (i)

{

case 1:

is\_find = TRUE;

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

ship\_res = D\_find(ship);

system("pause");

break;

case 2:

ship = D\_sort(ship);

system("pause");

break;

}

}

else

{

puts("Список пуст, обрабатывать нечего");

system("pause");

}

break;

case 5:

if (is\_find)

if (ship\_res != NULL)

D\_print(ship\_res);

else

puts("При последнем поиске не найдено элементов,\nсоответствующих заданным

критериям");

else

puts("Вы ещё ни разу не проводили поиск");

system("pause");

break;

case 6:

puts("Справка");

puts("При вводе строки:");

puts("1. таб конвертируется в пробел");

puts("2. пробелы сначала вводить запрещено");

puts("3. пробелы в конце убираются");

puts("Кораблик имеет пять полей:");

puts("1. Имя-строка от 1 до 30 символов");

puts("2. Тип-строка от 1 до 20 символов");

puts("3. Положение по x-целое число от -999999 до 9999999");

puts("4. Положение по y-целое число от -999999 до 9999999");

puts("5. Положение по z-целое число от -999999 до 9999999");

puts("Результат обработки - новый массив данных состоящий из элементов исходного");

puts("массива, которые соответствуют заданным критериям.");

puts("Русские буквы вводить запрещено.");

puts("Для двухсвязных списков возможен вывод с начала");

puts("и с конца");

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu != 0);

ship = D\_free\_all(ship);

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

}

//@@

typedef struct ST3

{

DATA info; //Данные о кораблике

struct ST3 \*link1;//Указатель на следующий элемент

struct ST3 \*link2;//Указатель на предыдующий элемент

} CSHIP;

//Функция для добавления списка в начало исходного списка

CSHIP \*C\_add\_frist(CSHIP \*ship, CSHIP \*sp)

{

if (ship != NULL)

{

CSHIP\* l;//Вспомогательный указатель

ship->link2->link1 = sp;

sp->link2->link1 = ship;

l = sp->link2;

sp->link2 = ship->link2;

ship->link2 = l;

}

return sp;

}

//Функция для добавления списка в начало исходного списка

CSHIP \*C\_add\_last(CSHIP \*ship, CSHIP \*sp)

{

if (ship != NULL)

{

CSHIP\* l;//Вспомогательный указатель

ship->link2->link1 = sp;

sp->link2->link1 = ship;

l = sp->link2;

sp->link2 = ship->link2;

ship->link2 = l;

}

else

ship = sp;

return ship;

}

//Функция для добавления списка в начало исходного списка

CSHIP \*C\_add\_after(CSHIP \*ship, CSHIP \*sp)

{

if (ship != NULL)

ship = C\_add\_frist(ship->link1,sp)->link2;

else

ship = sp;

return ship;

}

//Функция возвращает указатель на первый элемент списка

CSHIP \*C\_get\_structs()

{

CSHIP \*n\_sp; //Указатель на первый элемент списка

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

CSHIP \*new\_el; //Указатель на введённую структуру

bool is\_end\_input;//Указатель на введённую структуру

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

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

n\_sp->link1 = n\_sp;

n\_sp->link2 = n\_sp;

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

if (input\_yn())

{

is\_end\_input = false;

last\_el = n\_sp;

do

{

new\_el = (CSHIP\*)malloc(sizeof(CSHIP));

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

new\_el->link1 = new\_el;

new\_el->link2 = new\_el;

C\_add\_after(last\_el,new\_el);

last\_el = new\_el;

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

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

} while (!is\_end\_input);

}

return n\_sp;

}

//Функция для вычисления длинны списка

int C\_get\_len(CSHIP\*ship)

{

int i; //Вспомогательная переменная

CSHIP \*link;//Вспомогательная переменная

i = 0;

if (ship != NULL)

for (i = 1, link = ship->link1; link != ship; link = link->link1, i++);

return i;

}

//Функция для добавления элементов в список

CSHIP \*C\_add(CSHIP \*ship)

{

int menu; //Выбранный пункт меню

int n; //Вспомогательная переменная

int len; //Количество элементов в списке

CSHIP \*new\_sp;//Указатель на первый элемент списка

CSHIP \*link; //Вспомогательный указатель

puts("Вводите последовательность элементов");

if (ship != NULL)

{

new\_sp = C\_get\_structs();

do

{

system("cls");

puts("Вставка");

puts("1-Вставить в начало");

puts("2-Вставить в конец");

puts("3-Вставить после n-ного элемента");

menu = input\_number();

if (menu<1 || menu>4)

{

puts("Нет такого пункта");

system("pause");

}

} while (menu<1 || menu>4);

switch (menu)

{

case 1:

ship = C\_add\_frist(ship, new\_sp);

break;

case 2:

ship = C\_add\_last(ship, new\_sp);

break;

case 3:

len = C\_get\_len(ship);

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");

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

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

C\_add\_after(link, new\_sp);

break;

}

}

else

ship = C\_get\_structs();

puts("Новые элементы успешно добавлены");

return ship;

}

//Функция для удаления одного элемента из списка

CSHIP\* C\_del\_one(CSHIP \*link)

{

if (link != NULL)

if (link->link1 != link)

{

link->link1->link2 = link->link2;

link = link->link1;

free(link->link2->link1->info.name);

free(link->link2->link1->info.type);

free(link->link2->link1);

link->link2->link1 = link;

}

else

{

free(link->info.name);

free(link->info.type);

free(link);

link = NULL;

}

return link;

}

//Функция для удаления элементов из списка

CSHIP \*C\_del(CSHIP \*ship)

{

int len; //Количество элементов в списке

int m; //Номер последнего удаляемого элемента

int n; //Номер первого удаляемого элемента

int i; //Вспомогательная переменная

CSHIP \*link; //Вспомогательный указатель

if (ship != NULL)

{

len = C\_get\_len(ship);

puts("Удаление элементов с n до m");

do

{

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

n = input\_number();

if (n<1 || n>len)

puts("Неправильный ввод");

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

do

{

printf("Вводите m(от %i до %i)\n", n, len);

m = input\_number();

if (m<n || m>len)

puts("Неправильный ввод");

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

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

i = m - n + 1;

do

{

link=C\_del\_one(link);

i--;

} while (i > 0);

if (n == 1)

ship = link;

puts("Элемент(-ы) успешно удалены");

}

else

puts("Список пуст, удалять нечего");

return ship;

}

//Функция для вывода информационных полей одной структуры

void C\_print\_struct(CSHIP \*ship)

{

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

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

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

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

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

}

//Функция для вывода списка

void C\_print(CSHIP \*ship)

{

int i, r,a; //Вспомогательные переменные

int type; //Тип вывода(1-в одну сторону 2-в другую)

CSHIP \*link;//Вспомогательный указатель

int len; //Количество элементов в списке

int n; //Номер элемента, с которого начинается вывод

int linkadd;//Количество битов, которое надо прибавить к указателю на элемент списка,

чтобы получить адрес следующего выводимого элемента

if (ship != NULL)

{

do

{

puts("1-Вывод по часовой с n-ого элемента");

puts("2-Вывод против часовой с n-ого элемента");

type = input\_number();

if (type < 1 || type > 2)

puts("Нет такого пункта");

} while (type < 1 || type > 2);

len = C\_get\_len(ship);

do

{

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

n = input\_number();

if (n < 1 || n > len)

puts("Нет такого пункта");

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

for (i = 1; i != n; i++, ship = ship->link1);

printf("\nКораблик номер: %i\n", i);

C\_print\_struct(ship);

if (type == 1)

{

r = (n + 2) % 3;

linkadd = (char\*)&(ship->link1) - (char\*)ship;

a = 1;

}

else

{

i = i + len - 2;

r = (i + 1) % 3;

linkadd = (char\*)&(ship->link2) - (char\*)ship;

a = -1;

}

if (ship->link1 != ship)

{

link = \*((CSHIP\*\*)((char\*)ship + linkadd));

do

{

if (i % 3 == r)

system("pause");

printf("\nКораблик номер: %i\n", i % len + 1);

i += a;

C\_print\_struct(link);

link = \*((CSHIP\*\*)((char\*)link + linkadd));

} while (link != ship);

}

printf("\nВывод окончен\n");

}

else

puts("Список пуст");

}

//Функция для освобождения памяти из под списка

CSHIP\* C\_free\_all(CSHIP\* ship)

{

CSHIP \*link;//Вспомогательный указатель

if (ship != NULL)

{

ship->link2->link1 = NULL;

do

{

link = ship;

ship = ship->link1;

free(link->info.name);

free(link->info.type);

free(link);

} while (ship != NULL);

}

return NULL;

}

//Функция для создания копии элемента списка

CSHIP \*C\_create\_copy(CSHIP\* ship)

{

CSHIP \*new\_ship;//Указатель на копированный элемент

int i; //Указатель на копированный элемент

int len; //Количество элементов в списке

if (ship != NULL)

{

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

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

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

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

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 = new\_ship;

new\_ship->link2 = new\_ship;

}

else

new\_ship = NULL;

return new\_ship;

}

//Функция для вставления элемента в отсортированный список

CSHIP \*C\_insert(CSHIP \*ship, CSHIP \*el, SORTDATA\*sort)

{

if (ship != NULL)

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

{

CSHIP \*l;//Вспомогатеьный указатель

l = ship;

while (ship->link1 != l)

{

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

{

C\_add\_after(ship, el);

return l;

}

ship = ship->link1;

}

C\_add\_after(ship, el);

return l;

}

else

{

C\_add\_after(ship->link2,el);

return el;

}

else

{

el->link1 = el;

el->link2 = el;

return el;

}

}

//Функция для сортировки

CSHIP \*C\_sort(CSHIP \*ship)

{

if (ship == NULL)

{

puts("Нечего сортировать");

return NULL;

}

CSHIP \*ship\_sorted;//возвращает указатель на первый элемент списка

CSHIP \*link; //Вспомогательный указатель

SORTDATA sort; //Данные о сортировке

select\_sort(&sort);

ship\_sorted = NULL;

ship->link2->link1 = NULL;

do

{

link = ship;

ship = ship->link1;

link->link1 = link;

link->link2 = link;

ship\_sorted = C\_insert(ship\_sorted, link, &sort);

} while (ship != NULL);

puts("Сортировка окончена");

return ship\_sorted;

}

//Функция для поиска

CSHIP\* C\_find(CSHIP\* ship)

{

FINDDATA find; //Данные о поиске

CSHIP \*ship\_res;//Указатель на первый элемент обработанного списка

CSHIP \*link; //Вспомогательный указатель

CSHIP \*new\_el; //Вспомогательный указатель

select\_find(&find);

ship\_res = NULL;

link = ship;

ship->link2->link1 = NULL;

do

{

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

{

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

ship\_res = C\_add\_frist(ship\_res, new\_el)->link2;

}

ship = ship->link1;

} while (ship != NULL);

if (ship\_res != NULL)

puts("Поиск завершон");

else

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

link->link2->link1 = link;

return ship\_res;

}

//Функция для создания меню

void C\_menu()

{

int menu; //Выбранный пункт меню

CSHIP \*ship; //Указатель на первый элемент списка исходных данных

CSHIP \*ship\_res;//Указатель на первый элемент списка результатов поиска

bool is\_find; //Флажок поиска

int i; //Вспомогательная переменная

is\_find = false;

ship = NULL;

ship\_res = NULL;

do

{

system("cls");

puts("Кольцевые двухсвязные списки");

puts("1-добавление элементов в список");

puts("2-удаление элементов из списка");

puts("3-вывод элементов списка");

puts("4-обработка списка");

puts("5-вывод результатов последнего поиска");

puts("6-справка");

puts("0-возврат в надменю");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

ship = C\_add(ship);

system("pause");

break;

case 2:

ship = C\_del(ship);

system("pause");

break;

case 3:

C\_print(ship);

system("pause");

break;

case 4:

if (ship != NULL)

{

do

{

system("cls");

puts("Выберите тип обработки");

puts("1-поиск (при этом формируется новый список)");

puts("2-сортировка элементов исходного списка");

puts("0-возврат в надменю");

i = input\_number();

if (i<0 || i>2)

{

puts("Вы неправильно ввели пункт меню");

system("pause");

}

} while (i<0 || i>2);

switch (i)

{

case 1:

is\_find = TRUE;

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

ship\_res = C\_find(ship);

system("pause");

break;

case 2:

ship = C\_sort(ship);

system("pause");

break;

}

}

else

{

puts("Список пуст, обрабатывать нечего");

system("pause");

}

break;

case 5:

if (is\_find)

if (ship\_res != NULL)

C\_print(ship\_res);

else

puts("При последнем поиске не найдено элементов,\nсоответствующих заданным

критериям");

else

puts("Вы ещё ни разу не проводили поиск");

system("pause");

break;

case 6:

puts("Справка");

puts("При вводе строки:");

puts("1. таб конвертируется в пробел");

puts("2. пробелы сначала вводить запрещено");

puts("3. пробелы в конце убираются");

puts("Кораблик имеет пять полей:");

puts("1. Имя-строка от 1 до 30 символов");

puts("2. Тип-строка от 1 до 20 символов");

puts("3. Положение по x-целое число от -999999 до 9999999");

puts("4. Положение по y-целое число от -999999 до 9999999");

puts("5. Положение по z-целое число от -999999 до 9999999");

puts("Результат обработки - новый массив данных состоящий из элементов исходного");

puts("массива, которые соответствуют заданным критериям.");

puts("Русские буквы вводить запрещено.");

puts("Для двухсвязных кольцевых списков возможен вывод с n-ого элемента");

puts("по часовой и пртив часовой стрелки");

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu != 0);

ship = C\_free\_all(ship);

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

}

//@@

int main()

{

setlocale(0, "RU");

puts(" ООО БББББ ССС ААА");

puts("О О Б С А А");

puts("О О ББББ С ААААА");

puts("О О Б Б С А А");

puts(" ООО ББББ ССС А А");

system("pause");

char menu;//Выбранный пункт меню

do

{

system("cls");

puts("Списки");

puts("1-односвязные списки");

puts("2-двухсвязные списки");

puts("3-кольцевые двухсвязные списки");

puts("4-справка");

puts("0-выход");

menu = input\_number();

system("cls");

switch (menu)

{

case 1:

O\_menu();

break;

case 2:

D\_menu();

break;

case 3:

C\_menu();

break;

case 4:

puts("Эта программа обрабатывает заданный список данных о корабликах.");

puts("Эта программа работает со списками трёх типов: ");

puts("односвязный, двухсвязный, кольцевой двухсвязный.");

puts("Координаты в данной программе целочисленные.");

puts("При сравнении названий\\типов сравнивается сначала первый символ,");

puts("потом второй и.т.д.");

puts("При сравнении символов используется их коды.");

system("pause");

break;

case 0:

break;

default:

puts("Нет такого пункта меню");

system("pause");

break;

}

} while (menu != 0);

}

# 9.Результат

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

# Вывод

В результате выполнения лабораторной работы получены практические навыки работы со списками в языке C/C++.