Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новосибирский государственный технический университет»

Кафедра Автоматизированных систем управления

**Практическая работа**

по дисциплине «Программирование»

Тема: Списки и структуры

|  |  |
| --- | --- |
| Выполнили:  Кривощапов Сергей  Писарев Радомир  Пирожников Владислав  Группа АВТ-414  Факультет АВТФ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «15» марта 2025 г. | Проверил:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ф. И. О.)  Балл: \_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

Новосибирск 2025

***Определение структуры данных с комментариями о назначении полей***

|  |
| --- |
| typedef struct WIFI  {  char vendor[MAX\_VENDOR\_LENGTH]; // название бренда Wifi маршрутизатора  unsigned short port\_count; // количество Ethernet портов  enum \_5G\_MARK has\_5G; // наличие диапазона Wi-Fi 5 ГГц  }WIFI; |

***Прототипы разработанных функций (шаги 2 – 5). Комментарии об особенностях их работы (алгоритм).***

|  |
| --- |
| //Функция для инициализации структуры.  //В качестве аргументов принимает структуру WIFI по указателю, символьный массив brand\_name, целочисленную переменную port\_count и переменную перечислимого типа has\_5G.  int init\_struct(WIFI\* router, const char\* vendor, unsigned short port\_count, enum \_5G\_MARK has\_5G );  //Функция для вывода всех полей структуры.  //В качестве аргумента получает структуру по указателю.  void print\_struct(const WIFI\* router);  // Проверяет равенство двух экземпляров структуры посредством сравнения значений по каждому из полей экземпляров.  // Возвращает 1, если экземпляры идентичны.  int is\_equal(const WIFI\* router1, const WIFI\* router2);  // Функция для сравнения двух экземпляров структур. Возвращает 0, если "больше" первый экземпляр, 1 если второй соответственно.  // Приоритеты для сравнения определены: 1) vendor 2) port\_count, has\_5G.  // Если vendor у экземпляров совпадает, производится вызов compare\_int, которая сравнивает экземпляры по второму приоритету.  int compare(const WIFI\* router1, const WIFI\* router2); |

***Определение элемента списка с комментариями о назначении полей***

|  |
| --- |
| typedef struct WIFI\_item  {  WIFI data; // Данные о роутере по значению  struct WIFI\_item\* next; // Указатель на следующий элемент списка  }WIFI\_item;  typedef struct WIFI\_list  {  struct WIFI\_item\* head; // Указатель на первый элемент списка  }WIFI\_list; |

***Прототипы разработанных функций (шаги 7 – 9). Комментарии об особенностях их работы (алгоритм).***

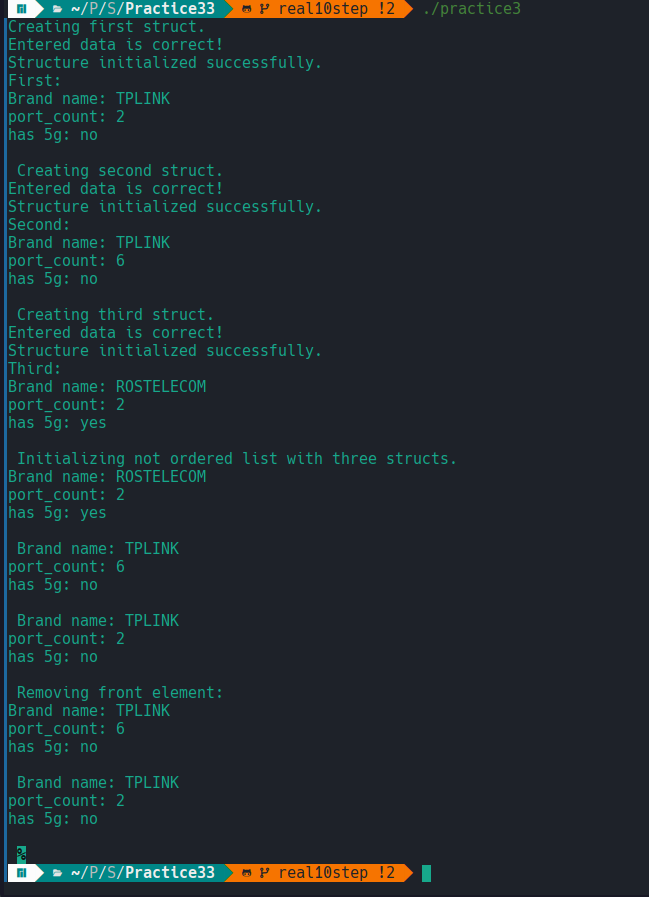
|  |
| --- |
| // Функция добавления элемента в начало списка  // Принимает в качестве аргументов список роутеров по указателю, экземпляр структуры роутера по значению.  // Выделяет память динамически для нового элемента.  // Устанавливаются новые значения указателей next, head  // Возвращает 0 при успешном исходе, иначе 1  int push\_front(WIFI\_list\* list, WIFI data);  // Функция удаления первого элемента списка  // Принимает в качестве аргумента список роутеров по указателю.  // обновляет head, освобождает память для удаленного элемента.  void delete\_front(WIFI\_list\* list);  // Функция вывода сведений для каждого элемента списка  // Принимает в качестве аргумента список роутеров по указателю.  // Внутри вызывает print\_struct  void print\_list(const WIFI\_list\* list); |

***Текст тестирующей функции (main), написанной на шаге 10 с комментариями.***

|  |
| --- |
| #include "WIFI.h"  #include "list.h"  int main()  {  // Инициализация трёх экземпляров структуры на стеке  printf("Creating first struct.\n");  WIFI router1;  if (struct\_init\_result(init\_struct(&router1, "TPLINK", 2, Is5G)) != 0) return 1;  printf("First: \n");  print\_struct(&router1); // Вывод сведений о структуре  printf("Creating second struct.\n");  WIFI router2;  if (struct\_init\_result(init\_struct(&router2, "TPLINK", 6, Is5G)) != 0) return 1;  printf("Second: \n");  print\_struct(&router2);  printf("Creating third struct.\n");  WIFI router3;  if (struct\_init\_result(init\_struct(&router3, "ROSTELECOM", 2, Not5G)) != 0) return 1;  printf("Third: \n");  print\_struct(&router3);  // Инициализация неупорядоченного списка  printf("Initializing not ordered list with three structs.\n");  WIFI\_list\* router\_list = (WIFI\_list\*)malloc(sizeof(WIFI\_list));  if (router\_list == NULL)  {  printf("Can't create list.\n");  return 1;  }  router\_list->head = NULL;    // Добавляем элементы в начало списка  if (push\_front(router\_list, router1) == 1) return 1;  if (push\_front(router\_list, router2) == 1) return 1;  if (push\_front(router\_list, router3) == 1) return 1;  print\_list(router\_list); // Вывод сведений о каждом элементе списка  printf("Removing front element:\n");  delete\_front(router\_list); // Удаления первого элемента списка    print\_list(router\_list);  delete\_list(router\_list); // Удаление списка полностью.  return 0;  } |

***Скриншот результата запуска тестовой функции и выводы о полученных результатах.***

Выводы: экземпляры структуры инициализируются, их сведения проверяются, выводятся корректно. Неупорядоченный список инициализируется корректно. Удаление элемента из начала списка, вывод списка успешны.



***Прототипы разработанных функций (шаги 11 – 12). Комментарии об особенностях их работы (алгоритм).***

|  |
| --- |
| // Добавляет новый элемент в список с сохранением условий упорядочивания. Список упорядочен по 1)vendor 2)port\_count, has\_5G  // Принимает в качестве аргументов список роутеров по указателю, экземпляр структуры роутера по значению.  // При успешной инициализации нового элемента возвращает возвращает 0, иначе 1  int add\_to\_ordered(WIFI\_list\* list, WIFI data);  // Функция удаления заданного элемента из списка  // Принимает в качестве аргументов список роутеров по указателю, экземпляр структуры роутера по значению.  // Находит элемент через is\_equal  // Если удаляемый элемент в начале списка обновляем head  // Если нет, обновляем next. Если соответствующий элемент отсутствует в списке, выводится ошибка и список остаётся прежним.  void delete\_selected\_element(WIFI\_list\* list, WIFI data); |

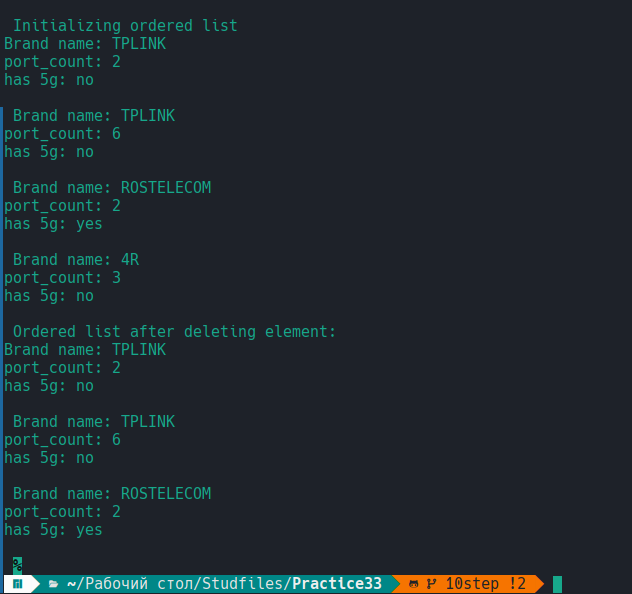
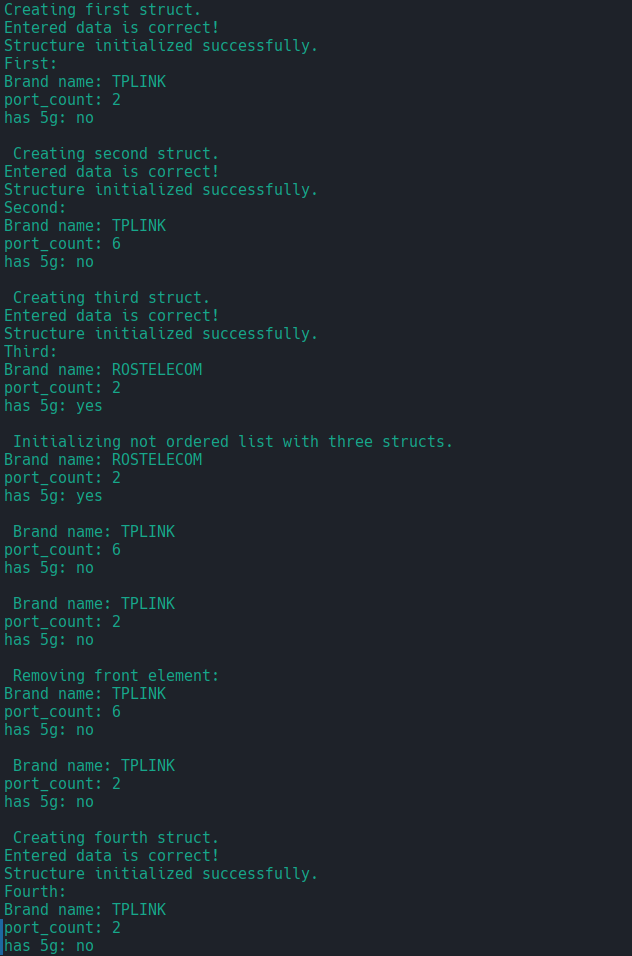
***Текст тестирующей функции (main), написанной на шаге 13 с комментариями.***

|  |
| --- |
| #include "WIFI.h"  #include "list.h"  int main()  {  // Инициализация трёх экземпляров структуры на стеке  printf("Creating first struct.\n");  WIFI router1;  if (struct\_init\_result(init\_struct(&router1, "TPLINK", 2, Is5G)) != 0) return 1;  printf("First: \n");  print\_struct(&router1); // Вывод сведений о структуре  printf("Creating second struct.\n");  WIFI router2;  if (struct\_init\_result(init\_struct(&router2, "TPLINK", 6, Is5G)) != 0) return 1;  printf("Second: \n");  print\_struct(&router2);  printf("Creating third struct.\n");  WIFI router3;  if (struct\_init\_result(init\_struct(&router3, "ROSTELECOM", 2, Not5G)) != 0) return 1;  printf("Third: \n");  print\_struct(&router3);  // Инициализация неупорядоченного списка  printf("Initializing not ordered list with three structs.\n");  WIFI\_list\* router\_list = (WIFI\_list\*)malloc(sizeof(WIFI\_list));  if (router\_list == NULL)  {  printf("Can't create list.\n");  return 1;  }  router\_list->head = NULL;    // Добавляем элементы в начало списка  if (push\_front(router\_list, router1) == 1) return 1;  if (push\_front(router\_list, router2) == 1) return 1;  if (push\_front(router\_list, router3) == 1) return 1;  print\_list(router\_list); // Вывод сведений о каждом элементе списка  printf("Removing front element:\n");  delete\_front(router\_list); // Удаления первого элемента списка    print\_list(router\_list);  delete\_list(router\_list); // Удаление списка полностью.    printf("Creating fourth struct.\n");  WIFI router4;  if (struct\_init\_result(init\_struct(&router4, "4R", 3, Is5G)) != 0) return 1;  printf("Fourth: \n");  print\_struct(&router1);  printf("Initializing ordered list\n");  if(add\_to\_ordered(router\_list, router1) == 1) return 1;  if(add\_to\_ordered(router\_list, router2) == 1) return 1;  if(add\_to\_ordered(router\_list, router3) == 1) return 1;  if(add\_to\_ordered(router\_list, router4) == 1) return 1;  print\_list(router\_list);  delete\_selected\_element(router\_list, router4);  printf("Ordered list after deleting element:\n");  print\_list(router\_list);  delete\_list(router\_list);  return 0;  } |

***Скриншот результата запуска тестовой функции и выводы о полученных результатах.***

Выводы: экземпляры структуры инициализируются, их сведения проверяются, выводятся корректно. Неупорядоченный список инициализируется корректно. Удаление элемента из начала списка, вывод списка успешны.

Упорядоченный список инициализирован корректно, сохранение порядка упорядочивания выполняется. Из упорядоченного списка удален выбранный элемент. Порядок упорядочивания списка сохранён.



***Определение элемента списка с комментариями о назначении полей, прототипы скорректированных функций.***

Определение элемента списка с комментариями о назначении полей

|  |
| --- |
| typedef struct WIFI\_item  {  WIFI\* data; // указатель на экземпляр структуры с данными о роутере  struct WIFI\_item\* next; // Указатель на следующий элемент списка  }WIFI\_item;  typedef struct WIFI\_list  {  struct WIFI\_item\* head; // Указатель на первый элемент списка  }WIFI\_list; |

Прототипы скорректированных функций

|  |
| --- |
| // Добавляет новый элемент в список с сохранением условий упорядочивания.  // Список упорядочен по 1)vendor 2)port\_count, has\_5G  // Принимает в качестве аргументов список роутеров по указателю, указатель на данные о роутере  // При успешной инициализации нового элемента возвращает возвращает 0, иначе 1  int add\_to\_ordered(WIFI\_list\* list, WIFI\* data);  // Функция удаления заданного элемента из списка  // Принимает в качестве аргументов список роутеров по указателю, указатель на данные о роутере.  // Находит элемент через is\_equal  // Если удаляемый элемент в начале списка обновляем head  // Если нет, обновляем next. Если соответствующий элемент отсутствует в списке, выводится ошибка и список остаётся прежним.  void delete\_selected\_element(WIFI\_list\* list, WIFI\* data); |

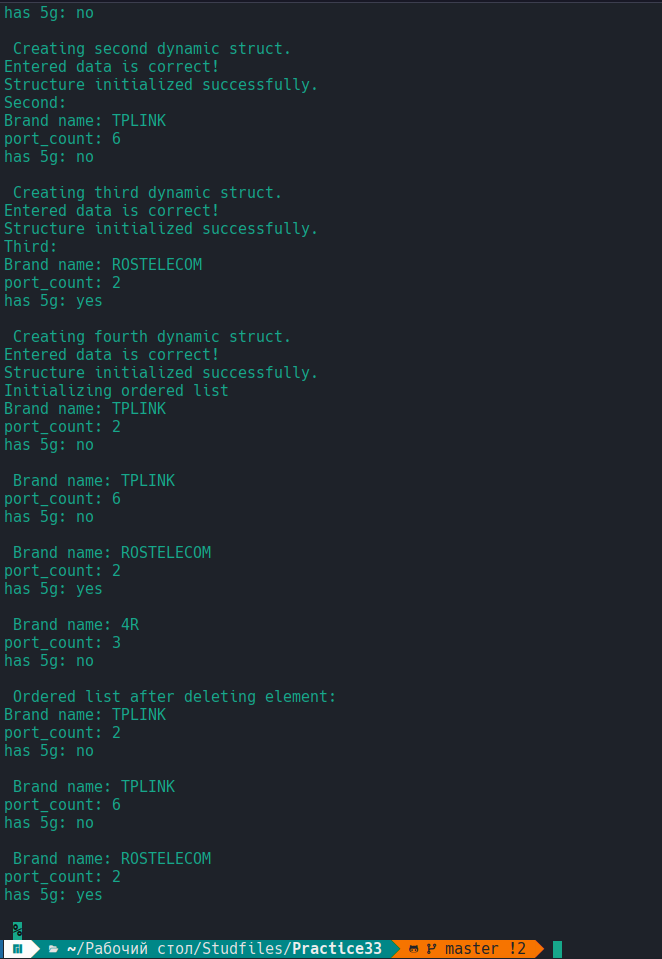
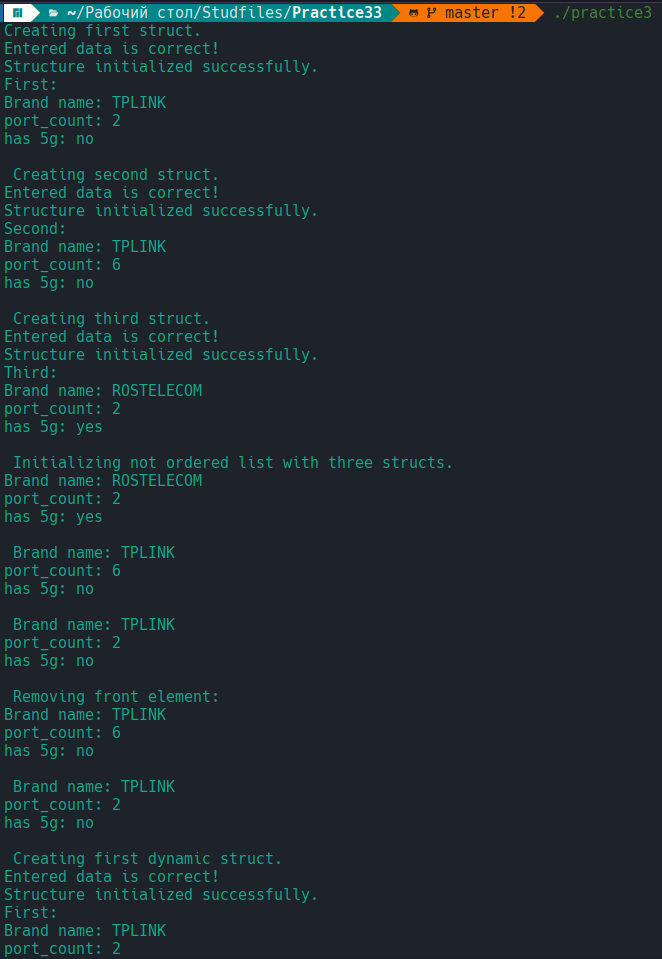
Текст тестирующей функции (main), написанной на шаге 14 с комментариями.

|  |
| --- |
| #include "WIFI.h"  #include "list.h"  int main()  {  // Инициализация в динамической памяти трёх экземпляров структуры  printf("Creating first struct.\n");  WIFI\* router1 = dynamic\_struct\_create("TPLINK", 2, Is5G);  if (router1 == NULL) return 1; // Проверка корректности инициализации экземпляра  printf("First: \n");  print\_struct(router1); // Вывод сведений о структуре  printf("Creating second struct.\n");  WIFI\* router2 = dynamic\_struct\_create("TPLINK", 6, Is5G);  if (router2 == NULL) return 1;    printf("Second: \n");  print\_struct(router2);  printf("Creating third struct.\n");  WIFI\* router3 = dynamic\_struct\_create("ROSTELECOM", 2, Not5G);    if (router3 == NULL) return 1;  printf("Third: \n");  print\_struct(router3);  // Инициализация неупорядоченного списка  printf("Initializing not ordered list with three structs.\n");  WIFI\_list\* router\_list = (WIFI\_list\*)malloc(sizeof(WIFI\_list));  if (router\_list == NULL)  {  printf("Can't create list.\n");  return 1;  }  router\_list->head = NULL;    // Добавляем элементы в начало списка  if (push\_front(router\_list, router1) == 1) return 1;  if (push\_front(router\_list, router2) == 1) return 1;  if (push\_front(router\_list, router3) == 1) return 1;  print\_list(router\_list); // Вывод сведений о каждом элементе списка  printf("Removing front element:\n");  delete\_front(router\_list); // Удаления первого элемента списка    print\_list(router\_list);  delete\_list(router\_list); // Удаление списка полностью.    //Освобождение динамической памяти, выделенной под экземпляры структуры  dynamic\_struct\_free(router1);  dynamic\_struct\_free(router2);  dynamic\_struct\_free(router3);    printf("Creating first dynamic struct.\n");  router1 = dynamic\_struct\_create("TPLINK", 2, Is5G);  if (router1 == NULL) return 1;  printf("First: \n");  print\_struct(router1);  printf("Creating second dynamic struct.\n");  router2 = dynamic\_struct\_create("TPLINK", 6, Is5G);  if (router2 == NULL) return 1;    printf("Second: \n");  print\_struct(router2);  printf("Creating third dynamic struct.\n");  router3 = dynamic\_struct\_create("ROSTELECOM", 2, Not5G);    if (router3 == NULL) return 1;  printf("Third: \n");  print\_struct(router3);  printf("Creating fourth dynamic struct.\n");  WIFI\* router4 = dynamic\_struct\_create("4R", 3, Is5G);  if (router4 == NULL) return 1;    printf("Initializing ordered list\n");  if(add\_to\_ordered(router\_list, router1) == 1) return 1;  if(add\_to\_ordered(router\_list, router2) == 1) return 1;  if(add\_to\_ordered(router\_list, router3) == 1) return 1;  if(add\_to\_ordered(router\_list, router4) == 1) return 1;  print\_list(router\_list);  delete\_selected\_element(router\_list, router4);  printf("Ordered list after deleting element:\n");  print\_list(router\_list);  delete\_list(router\_list);  dynamic\_struct\_free(router1);  dynamic\_struct\_free(router2);  dynamic\_struct\_free(router3);  dynamic\_struct\_free(router4);  return 0;  } |

***Скриншот результата запуска тестовой функции и выводы о полученных результатах.***

Выводы: динамические экземпляры структуры инициализируются, их сведения проверяются, выводятся корректно. Неупорядоченный список инициализируется корректно. В узлах списка хранятся указатели на экземпляры структуры с данными о роутерах. Удаление элемента из начала списка, вывод списка успешны.

Упорядоченный список инициализирован корректно, сохранение порядка упорядочивания выполняется. Из упорядоченного списка удален выбранный элемент. Порядок упорядочивания списка сохранён.



***Исходные коды программ***

***После выполнения шага 14***

***WIFI.h***

|  |
| --- |
| ***#include <stddef.h>***  ***#include <stdio.h>***  ***#include <string.h>***  ***#include <stdlib.h>***  ***#include <limits.h>***  ***#include <math.h>***  ***#ifndef WIFI\_H***  ***#define WIFI\_H***  ***#define MAX\_VENDOR\_LENGTH 20***  ***#define MAX\_INT\_LENGTH 8***  ***#define MAX\_MARK\_LENGTH 10***  ***enum ERROR\_CODE***  ***{***  ***Success, Invalid\_Port\_Count, Invalid\_5G\_Mark***  ***};***  ***enum \_5G\_MARK***  ***{***  ***Is5G, Not5G, Undefined***  ***};***  ***typedef struct WIFI***  ***{***  ***char vendor[MAX\_VENDOR\_LENGTH]; // название бренда Wifi маршрутизатора***  ***unsigned short port\_count; // количество ethernet портов***  ***enum \_5G\_MARK has\_5G; // наличие диапазона Wi-Fi 5 ГГц***  ***}WIFI;***  ***int input\_int(const char\* message);***  ***void input\_string(char\* buffer, int buffer\_size);***  ***enum \_5G\_MARK input\_5G\_mark(const char\* message);***  ***enum ERROR\_CODE router\_DataCheck(const WIFI\* router);***  ***//Функция для инициализации структуры.***  ***//В качестве аргументов принимает структуру WIFI по указателю, символьный массив brand\_name, целочисленную переменную port\_count и переменную перечислимого типа has\_5G.***  ***int init\_struct(WIFI\* router, const char\* vendor, unsigned short port\_count, enum \_5G\_MARK has\_5G );***  ***int struct\_init\_result(const enum ERROR\_CODE result\_code);***  ***int init\_userdata(WIFI\* router);***  ***void copy\_struct(const WIFI\* router, WIFI\* copy);***  ***//Функция для вывода всех полей структуры.***  ***//В качестве аргумента получает структуру по указателю.***  ***void print\_struct(const WIFI\* router);***  ***int field\_changing(WIFI\* router);***  ***int update\_port\_count(WIFI\* router, const unsigned short value);***  ***int update\_vendor\_name(WIFI\* router, const char\* value);***  ***int update\_5G\_mark(WIFI\* router, const enum \_5G\_MARK value);***  ***WIFI\* dynamic\_struct\_create(const char\* vendor, const unsigned short port\_count, const enum \_5G\_MARK has\_5G);***  ***WIFI\* dynamic\_struct\_free(WIFI\* dym\_struct\_ptr);***  ***// Проверяет равенство двух экземпляров структуры посредством сравнения значений по каждому из полей экземпляров.***  ***// Возвращает 1, если экземпляры идентичны.***  ***int is\_equal(const WIFI\* router1, const WIFI\* router2);***  ***// Функция для сравнения двух экземпляров структур. Возвращает 0, если "больше" первый экземпляр, 1 если второй соответственно.***  ***// Приоритеты для сравнения определены: 1) vendor 2) port\_count, has\_5G.***  ***// Если vendor у экземпляров совпадает, производится вызов compare\_int, которая сравнивает экземпляры по второму приоритету.***  ***int compare(const WIFI\* router1, const WIFI\* router2);***  ***int compare\_int(const WIFI\* router1, const WIFI\* router2);***  ***#endif /\* WIFI\_H \*/*** |

***WIFI.c***

|  |
| --- |
| ***#define \_CRT\_SECURE\_NO\_WARNINGS***  ***#include "WIFI.h"***  ***#ifndef WIFI\_C***  ***#define WIFI\_C***  ***int input\_int(const char\* message)***  ***{***  ***char buffer[MAX\_INT\_LENGTH];***  ***char temp[MAX\_INT\_LENGTH];***  ***printf("%s", message);***  ***input\_string(buffer, MAX\_INT\_LENGTH);***  ***snprintf(temp, MAX\_INT\_LENGTH, "%d", atoi(buffer));***  ***if ((atoi(buffer) == 0 && buffer[0] != '0') || (atoi(buffer) >= USHRT\_MAX) || (atoi(buffer) < 0) || (strcmp(buffer, temp) != 0))***  ***{***  ***printf("Entered value is incorrect.");***  ***exit(EXIT\_FAILURE);***  ***}***  ***return atoi(buffer);***  ***}***  ***void input\_string(char\* buffer, int buffer\_size)***  ***{***  ***fgets(buffer, buffer\_size, stdin);***  ***size\_t len = strlen(buffer);***  ***if (len > 1 && buffer[len - 1] == '\n') buffer[len - 1] = '\0';***  ***else***  ***{***  ***printf("Entered value is too long or empty.\n");***  ***exit(EXIT\_FAILURE);***  ***}***  ***}***  ***enum \_5G\_MARK input\_5G\_mark(const char\* message)***  ***{***  ***char buffer[MAX\_MARK\_LENGTH];***  ***printf("%s", message);***  ***input\_string(buffer, MAX\_MARK\_LENGTH);***  ***if (strcmp(buffer, "yes") == 0) return Is5G;***  ***if (strcmp(buffer, "no") == 0) return Not5G;***  ***return Undefined;***  ***}***  ***enum ERROR\_CODE router\_DataCheck(const WIFI\* router)***  ***{***  ***if (router->port\_count > 32)***  ***{***  ***printf("Entered port count information is incorrect. Port count cant be more than 32.\n");***  ***return Invalid\_Port\_Count;***  ***}***  ***if(router->has\_5G == Undefined)***  ***{***  ***printf("Entered 5G mark is incorrect.");***  ***return Invalid\_5G\_Mark;***  ***}***  ***printf("Entered data is correct!\n");***  ***return Success;***  ***}***  ***int init\_struct(WIFI\* router, const char\* vendor, unsigned short port\_count, enum \_5G\_MARK has\_5G )***  ***{***  ***strncpy(router->vendor, vendor, MAX\_VENDOR\_LENGTH);***  ***router->vendor[MAX\_VENDOR\_LENGTH - 1] = '\0';***  ***router->port\_count = port\_count;***  ***router->has\_5G = has\_5G;***  ***return router\_DataCheck(router);***  ***}***  ***int struct\_init\_result(const enum ERROR\_CODE result\_code)***  ***{***  ***switch (result\_code)***  ***{***  ***case Success:***  ***printf("Structure initialized successfully.\n");***  ***return 0;***  ***default:***  ***printf("Structure initialization failed.\n");***  ***return 1;***  ***}***  ***}***  ***int init\_userdata(WIFI\* router)***  ***{***  ***char vendor[MAX\_VENDOR\_LENGTH];***  ***printf("Enter brand name: ");***  ***input\_string(vendor, MAX\_VENDOR\_LENGTH);***  ***unsigned short port\_count = input\_int("\nEnter port count: ");***  ***enum \_5G\_MARK has\_5G = input\_5G\_mark("\nHas 5G? (yes/no): ");***    ***if (struct\_init\_result(init\_struct(router, vendor, port\_count, has\_5G)) != 0) return 1;***  ***return 0;***  ***}***  ***void copy\_struct(const WIFI\* router, WIFI\* copy)***  ***{***  ***strcpy(copy->vendor, router->vendor);***  ***copy->port\_count = router->port\_count;***  ***copy->has\_5G = router->has\_5G;***  ***}***  ***void print\_struct(const WIFI\* router)***  ***{***  ***printf("Brand name: %s \nport\_count: %d\nhas 5g: %s\n\n ",***  ***router->vendor,***  ***router->port\_count,***  ***router->has\_5G ? "yes":"no");***  ***}***  ***int field\_changing(WIFI\* router)***  ***{***  ***int field\_tochange = input\_int("\nWhich field you want to change?\n1 - brand name\n2 - port count\n3 - Wifi 5G Mark\nEnter any other symbol if you won't change anything\n\nEnter only digit: ");***  ***short init\_checkmark;***  ***switch (field\_tochange)***  ***{***  ***case 1:***  ***char vendor\_name\_tochange[MAX\_VENDOR\_LENGTH];***  ***printf("\nEnter new vendor value: ");***  ***input\_string(vendor\_name\_tochange, MAX\_VENDOR\_LENGTH);***  ***init\_checkmark = struct\_init\_result(update\_vendor\_name(router, vendor\_name\_tochange));***  ***break;***  ***case 2:***  ***unsigned short port\_count\_tochange = input\_int("\nEnter new port count value: ");***  ***init\_checkmark = struct\_init\_result(update\_port\_count(router, port\_count\_tochange));***  ***break;***  ***case 3:***  ***unsigned short \_5G\_mark\_tochange = input\_5G\_mark("\nEnter new 5G mark value: ");***  ***init\_checkmark = struct\_init\_result(update\_5G\_mark(router, \_5G\_mark\_tochange));***  ***break;***  ***default:***  ***printf("\nUndefined value received. Nothing will be changed.\n");***  ***break;***  ***}***  ***if (init\_checkmark == 0) return 0;***  ***return 1;***  ***}***  ***int update\_port\_count(WIFI\* router, const unsigned short value)***  ***{***  ***router->port\_count = value;***  ***return router\_DataCheck(router);***  ***}***  ***int update\_vendor\_name(WIFI\* router, const char\* value)***  ***{***  ***strncpy(router->vendor, value, MAX\_VENDOR\_LENGTH);***  ***return router\_DataCheck(router);***  ***}***  ***int update\_5G\_mark(WIFI\* router, const enum \_5G\_MARK value)***  ***{***  ***router->has\_5G = value;***  ***return router\_DataCheck(router);***  ***}***  ***WIFI\* dynamic\_struct\_create(const char\* vendor, const unsigned short port\_count, const enum \_5G\_MARK has\_5G)***  ***{***  ***WIFI\* router = (WIFI\*)malloc(sizeof(WIFI));***    ***if (!router) return NULL;***  ***if (struct\_init\_result(init\_struct(router, vendor, port\_count, has\_5G)) == 0) return router;***  ***dynamic\_struct\_free(router);***  ***return NULL;***  ***}***  ***WIFI\* dynamic\_struct\_free(WIFI\* dym\_struct\_ptr)***  ***{***  ***free(dym\_struct\_ptr);***  ***dym\_struct\_ptr = NULL;***  ***}***  ***int is\_equal(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***return ((strcmp(router1->vendor, router2->vendor) == 0) && router1->port\_count == router2->port\_count && router1->has\_5G == router2->has\_5G);***  ***}***  ***int compare(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***int vend\_cmp\_result = strcmp(router2->vendor, router1->vendor);***  ***if (vend\_cmp\_result != 0) return (vend\_cmp\_result < 0);***  ***return compare\_int(router1, router2);***  ***}***  ***// Port\_count - Second priority of comparsion. 5G mark - third.***  ***// Returns 1 if p2 is bigger. Else 0.***  ***int compare\_int(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***if (router2->port\_count != router1->port\_count)***  ***{***  ***if ((router2->port\_count > router1->port\_count)) return 1;***  ***return 0;***  ***}***  ***if (router2->has\_5G > router1->has\_5G) return 1;***  ***return 0;***  ***}***  ***#endif*** |

***list.h***

|  |
| --- |
| ***#include "WIFI.h"***  ***#ifndef LIST\_H***  ***#define LIST\_H***  ***typedef struct WIFI\_item***  ***{***  ***WIFI\* data; // указатель на экземпляр структуры с данными о роутере***  ***struct WIFI\_item\* next; // Указатель на следующий элемент списка***  ***}WIFI\_item;***  ***typedef struct WIFI\_list***  ***{***  ***struct WIFI\_item\* head; // Указатель на первый элемент списка***  ***}WIFI\_list;***  ***// Функция добавления элемента в начало списка***  ***// Принимает в качестве аргументов список роутеров по указателю, указатель на данные о роутере.***  ***// Выделяет память динамически для нового элемента.***  ***// Устанавливаются новые значения указателей next, head***  ***// Возвращает 0 при успешном исходе, иначе 1***  ***int push\_front(WIFI\_list\* list, WIFI\* data);***  ***// Функция удаления первого элемента списка***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// обновляет head, освобождает память для удаленного элемента.***  ***void delete\_front(WIFI\_list\* list);***  ***// Функция вывода сведений для каждого элемента списка***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// Внутри вызывает print\_struct***  ***void print\_list(const WIFI\_list\* list);***  ***// Функция очищения списка.***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// вызывает delete\_front, пока в списке есть элементы***  ***void delete\_list(WIFI\_list\* list);***  ***// Добавляет новый элемент в список с сохранением условий упорядочивания.***  ***// Список упорядочен по 1)vendor 2)port\_count, has\_5G***  ***// Принимает в качестве аргументов список роутеров по указателю, указатель на данные о роутере***  ***// При успешной инициализации нового элемента возвращает возвращает 0, иначе 1***  ***int add\_to\_ordered(WIFI\_list\* list, WIFI\* data);***  ***// Функция удаления заданного элемента из списка***  ***// Принимает в качестве аргументов список роутеров по указателю, указатель на данные о роутере.***  ***// Находит элемент через is\_equal***  ***// Если удаляемый элемент в начале списка обновляем head***  ***// Если нет, обновляем next. Если соответствующий элемент отсутствует в списке, выводится ошибка и список остаётся прежним.***  ***void delete\_selected\_element(WIFI\_list\* list, WIFI\* data);***  ***#endif*** |

***list.c***

|  |
| --- |
| ***#include "list.h"***  ***int push\_front(WIFI\_list\* list, WIFI\* data)***  ***{***  ***WIFI\_item\* new\_item = (WIFI\_item\*)malloc(sizeof(WIFI\_item));***  ***if (!new\_item)***  ***{***  ***printf("Can't create new item");***  ***return 1;***  ***}***  ***new\_item->data = data;***  ***new\_item->next = list->head;***  ***list->head = new\_item;***  ***return 0;***  ***}***  ***void delete\_front(WIFI\_list\* list)***  ***{***  ***if (list->head == NULL)***  ***{***  ***printf("Empty.\n");***  ***return;***  ***}***  ***WIFI\_item\* temp = list->head;***  ***list->head = list->head->next;***  ***free(temp);***  ***}***  ***void print\_list(const WIFI\_list\* list)***  ***{***  ***WIFI\_item\* current = list->head;***  ***while (current != NULL)***  ***{***  ***print\_struct(current->data);***  ***current = current->next;***  ***}***  ***}***  ***void delete\_list(WIFI\_list\* list)***  ***{***  ***while (list->head != NULL) delete\_front(list);***  ***}***  ***int add\_to\_ordered(WIFI\_list\* list, WIFI\* data)***  ***{***  ***WIFI\_item\* new\_item = (WIFI\_item\*)malloc(sizeof(WIFI\_item));***  ***if (!new\_item)***  ***{***  ***printf("Can't create new item.");***  ***return 1;***  ***}***  ***new\_item->data = data;***  ***new\_item->next = NULL;***  ***if (list->head == NULL || compare(data, list->head->data) < 0)***  ***{***  ***new\_item->next = list->head;***  ***list->head = new\_item;***  ***}***  ***else***  ***{***  ***WIFI\_item\* current = list->head;***  ***while (current->next != NULL && compare(data, current->next->data) >= 0) current = current->next; //!!!!!***  ***new\_item->next = current->next;***  ***current->next = new\_item;***  ***}***  ***return 0;***  ***}***  ***void delete\_selected\_element(WIFI\_list\* list, WIFI\* data)***  ***{***  ***WIFI\_item\* current = list->head;***  ***WIFI\_item\* previous = NULL;***  ***while (current != NULL)***  ***{***  ***if (is\_equal(current->data, data))***  ***{***  ***if (previous == NULL) list->head = current->next;***  ***else previous->next = current->next;***  ***free(current);***  ***return;***  ***}***  ***previous = current;***  ***current = current->next;***  ***}***  ***printf("Selected element not in list.\n");***  ***}*** |

***main.c***

|  |
| --- |
| ***#include "WIFI.h"***  ***#include "list.h"***  ***int main()***  ***{***  ***// Инициализация в динамической памяти трёх экземпляров структуры***  ***printf("Creating first struct.\n");***  ***WIFI\* router1 = dynamic\_struct\_create("TPLINK", 2, Is5G);***  ***if (router1 == NULL) return 1; // Проверка корректности инициализации экземпляра***  ***printf("First: \n");***  ***print\_struct(router1); // Вывод сведений о структуре***  ***printf("Creating second struct.\n");***  ***WIFI\* router2 = dynamic\_struct\_create("TPLINK", 6, Is5G);***  ***if (router2 == NULL) return 1;***    ***printf("Second: \n");***  ***print\_struct(router2);***  ***printf("Creating third struct.\n");***  ***WIFI\* router3 = dynamic\_struct\_create("ROSTELECOM", 2, Not5G);***    ***if (router3 == NULL) return 1;***  ***printf("Third: \n");***  ***print\_struct(router3);***  ***// Инициализация неупорядоченного списка***  ***printf("Initializing not ordered list with three structs.\n");***  ***WIFI\_list\* router\_list = (WIFI\_list\*)malloc(sizeof(WIFI\_list));***  ***if (router\_list == NULL)***  ***{***  ***printf("Can't create list.\n");***  ***return 1;***  ***}***  ***router\_list->head = NULL;***    ***// Добавляем элементы в начало списка***  ***if (push\_front(router\_list, router1) == 1) return 1;***  ***if (push\_front(router\_list, router2) == 1) return 1;***  ***if (push\_front(router\_list, router3) == 1) return 1;***  ***print\_list(router\_list); // Вывод сведений о каждом элементе списка***  ***printf("Removing front element:\n");***  ***delete\_front(router\_list); // Удаления первого элемента списка***    ***print\_list(router\_list);***  ***delete\_list(router\_list); // Удаление списка полностью.***    ***//Освобождение динамической памяти, выделенной под экземпляры структуры***  ***dynamic\_struct\_free(router1);***  ***dynamic\_struct\_free(router2);***  ***dynamic\_struct\_free(router3);***    ***printf("Creating first dynamic struct.\n");***  ***router1 = dynamic\_struct\_create("TPLINK", 2, Is5G);***  ***if (router1 == NULL) return 1;***  ***printf("First: \n");***  ***print\_struct(router1);***  ***printf("Creating second dynamic struct.\n");***  ***router2 = dynamic\_struct\_create("TPLINK", 6, Is5G);***  ***if (router2 == NULL) return 1;***    ***printf("Second: \n");***  ***print\_struct(router2);***  ***printf("Creating third dynamic struct.\n");***  ***router3 = dynamic\_struct\_create("ROSTELECOM", 2, Not5G);***    ***if (router3 == NULL) return 1;***  ***printf("Third: \n");***  ***print\_struct(router3);***  ***printf("Creating fourth dynamic struct.\n");***  ***WIFI\* router4 = dynamic\_struct\_create("4R", 3, Is5G);***  ***if (router4 == NULL) return 1;***    ***printf("Initializing ordered list\n");***  ***if(add\_to\_ordered(router\_list, router1) == 1) return 1;***  ***if(add\_to\_ordered(router\_list, router2) == 1) return 1;***  ***if(add\_to\_ordered(router\_list, router3) == 1) return 1;***  ***if(add\_to\_ordered(router\_list, router4) == 1) return 1;***  ***print\_list(router\_list);***  ***delete\_selected\_element(router\_list, router4);***  ***printf("Ordered list after deleting element:\n");***  ***print\_list(router\_list);***  ***delete\_list(router\_list);***  ***dynamic\_struct\_free(router1);***  ***dynamic\_struct\_free(router2);***  ***dynamic\_struct\_free(router3);***  ***dynamic\_struct\_free(router4);***  ***return 0;***  ***}*** |

***После выполнения шага 13***

***WIFI.h***

|  |
| --- |
| ***#include <stddef.h>***  ***#include <stdio.h>***  ***#include <string.h>***  ***#include <stdlib.h>***  ***#include <limits.h>***  ***#include <math.h>***  ***#ifndef WIFI\_H***  ***#define WIFI\_H***  ***#define MAX\_VENDOR\_LENGTH 20***  ***#define MAX\_INT\_LENGTH 8***  ***#define MAX\_MARK\_LENGTH 10***  ***enum ERROR\_CODE***  ***{***  ***Success, Invalid\_Port\_Count, Invalid\_5G\_Mark***  ***};***  ***enum \_5G\_MARK***  ***{***  ***Is5G, Not5G, Undefined***  ***};***  ***typedef struct WIFI***  ***{***  ***char vendor[MAX\_VENDOR\_LENGTH]; // название бренда Wifi маршрутизатора***  ***unsigned short port\_count; // количество ethernet портов***  ***enum \_5G\_MARK has\_5G; // наличие диапазона Wi-Fi 5 ГГц***  ***}WIFI;***  ***int input\_int(const char\* message);***  ***void input\_string(char\* buffer, int buffer\_size);***  ***enum \_5G\_MARK input\_5G\_mark(const char\* message);***  ***enum ERROR\_CODE router\_DataCheck(const WIFI\* router);***  ***//Функция для инициализации структуры.***  ***//В качестве аргументов принимает структуру WIFI по указателю, символьный массив brand\_name, целочисленную переменную port\_count и переменную перечислимого типа has\_5G.***  ***int init\_struct(WIFI\* router, const char\* vendor, unsigned short port\_count, enum \_5G\_MARK has\_5G );***  ***int struct\_init\_result(const enum ERROR\_CODE result\_code);***  ***int init\_userdata(WIFI\* router);***  ***void copy\_struct(const WIFI\* router, WIFI\* copy);***  ***//Функция для вывода всех полей структуры.***  ***//В качестве аргумента получает структуру по указателю.***  ***void print\_struct(const WIFI\* router);***  ***int field\_changing(WIFI\* router);***  ***int update\_port\_count(WIFI\* router, const unsigned short value);***  ***int update\_vendor\_name(WIFI\* router, const char\* value);***  ***int update\_5G\_mark(WIFI\* router, const enum \_5G\_MARK value);***  ***WIFI\* dynamic\_struct\_create(const char\* vendor, const unsigned short port\_count, const enum \_5G\_MARK has\_5G);***  ***WIFI\* dynamic\_struct\_free(WIFI\* dym\_struct\_ptr);***  ***// Проверяет равенство двух экземпляров структуры посредством сравнения значений по каждому из полей экземпляров.***  ***// Возвращает 1, если экземпляры идентичны.***  ***int is\_equal(const WIFI\* router1, const WIFI\* router2);***  ***// Функция для сравнения двух экземпляров структур. Возвращает 0, если "больше" первый экземпляр, 1 если второй соответственно.***  ***// Приоритеты для сравнения определены: 1) vendor 2) port\_count, has\_5G.***  ***// Если vendor у экземпляров совпадает, производится вызов compare\_int, которая сравнивает экземпляры по второму приоритету.***  ***int compare(const WIFI\* router1, const WIFI\* router2);***  ***int compare\_int(const WIFI\* router1, const WIFI\* router2);***  ***#endif /\* WIFI\_H \*/*** |

***WIFI.c***

|  |
| --- |
| ***#define \_CRT\_SECURE\_NO\_WARNINGS***  ***#include "WIFI.h"***  ***#ifndef WIFI\_C***  ***#define WIFI\_C***  ***int input\_int(const char\* message)***  ***{***  ***char buffer[MAX\_INT\_LENGTH];***  ***char temp[MAX\_INT\_LENGTH];***  ***printf("%s", message);***  ***input\_string(buffer, MAX\_INT\_LENGTH);***  ***snprintf(temp, MAX\_INT\_LENGTH, "%d", atoi(buffer));***  ***if ((atoi(buffer) == 0 && buffer[0] != '0') || (atoi(buffer) >= USHRT\_MAX) || (atoi(buffer) < 0) || (strcmp(buffer, temp) != 0))***  ***{***  ***printf("Entered value is incorrect.");***  ***exit(EXIT\_FAILURE);***  ***}***  ***return atoi(buffer);***  ***}***  ***void input\_string(char\* buffer, int buffer\_size)***  ***{***  ***fgets(buffer, buffer\_size, stdin);***  ***size\_t len = strlen(buffer);***  ***if (len > 1 && buffer[len - 1] == '\n') buffer[len - 1] = '\0';***  ***else***  ***{***  ***printf("Entered value is too long or empty.\n");***  ***exit(EXIT\_FAILURE);***  ***}***  ***}***  ***enum \_5G\_MARK input\_5G\_mark(const char\* message)***  ***{***  ***char buffer[MAX\_MARK\_LENGTH];***  ***printf("%s", message);***  ***input\_string(buffer, MAX\_MARK\_LENGTH);***  ***if (strcmp(buffer, "yes") == 0) return Is5G;***  ***if (strcmp(buffer, "no") == 0) return Not5G;***  ***return Undefined;***  ***}***  ***enum ERROR\_CODE router\_DataCheck(const WIFI\* router)***  ***{***  ***if (router->port\_count > 32)***  ***{***  ***printf("Entered port count information is incorrect. Port count cant be more than 32.\n");***  ***return Invalid\_Port\_Count;***  ***}***  ***if(router->has\_5G == Undefined)***  ***{***  ***printf("Entered 5G mark is incorrect.");***  ***return Invalid\_5G\_Mark;***  ***}***  ***printf("Entered data is correct!\n");***  ***return Success;***  ***}***  ***int init\_struct(WIFI\* router, const char\* vendor, unsigned short port\_count, enum \_5G\_MARK has\_5G )***  ***{***  ***strncpy(router->vendor, vendor, MAX\_VENDOR\_LENGTH);***  ***router->vendor[MAX\_VENDOR\_LENGTH - 1] = '\0';***  ***router->port\_count = port\_count;***  ***router->has\_5G = has\_5G;***  ***return router\_DataCheck(router);***  ***}***  ***int struct\_init\_result(const enum ERROR\_CODE result\_code)***  ***{***  ***switch (result\_code)***  ***{***  ***case Success:***  ***printf("Structure initialized successfully.\n");***  ***return 0;***  ***default:***  ***printf("Structure initialization failed.\n");***  ***return 1;***  ***}***  ***}***  ***int init\_userdata(WIFI\* router)***  ***{***  ***char vendor[MAX\_VENDOR\_LENGTH];***  ***printf("Enter brand name: ");***  ***input\_string(vendor, MAX\_VENDOR\_LENGTH);***  ***unsigned short port\_count = input\_int("\nEnter port count: ");***  ***enum \_5G\_MARK has\_5G = input\_5G\_mark("\nHas 5G? (yes/no): ");***    ***if (struct\_init\_result(init\_struct(router, vendor, port\_count, has\_5G)) != 0) return 1;***  ***return 0;***  ***}***  ***void copy\_struct(const WIFI\* router, WIFI\* copy)***  ***{***  ***strcpy(copy->vendor, router->vendor);***  ***copy->port\_count = router->port\_count;***  ***copy->has\_5G = router->has\_5G;***  ***}***  ***void print\_struct(const WIFI\* router)***  ***{***  ***printf("Brand name: %s \nport\_count: %d\nhas 5g: %s\n\n ",***  ***router->vendor,***  ***router->port\_count,***  ***router->has\_5G ? "yes":"no");***  ***}***  ***int field\_changing(WIFI\* router)***  ***{***  ***int field\_tochange = input\_int("\nWhich field you want to change?\n1 - brand name\n2 - port count\n3 - Wifi 5G Mark\nEnter any other symbol if you won't change anything\n\nEnter only digit: ");***  ***short init\_checkmark;***  ***switch (field\_tochange)***  ***{***  ***case 1:***  ***char vendor\_name\_tochange[MAX\_VENDOR\_LENGTH];***  ***printf("\nEnter new vendor value: ");***  ***input\_string(vendor\_name\_tochange, MAX\_VENDOR\_LENGTH);***  ***init\_checkmark = struct\_init\_result(update\_vendor\_name(router, vendor\_name\_tochange));***  ***break;***  ***case 2:***  ***unsigned short port\_count\_tochange = input\_int("\nEnter new port count value: ");***  ***init\_checkmark = struct\_init\_result(update\_port\_count(router, port\_count\_tochange));***  ***break;***  ***case 3:***  ***unsigned short \_5G\_mark\_tochange = input\_5G\_mark("\nEnter new 5G mark value: ");***  ***init\_checkmark = struct\_init\_result(update\_5G\_mark(router, \_5G\_mark\_tochange));***  ***break;***  ***default:***  ***printf("\nUndefined value received. Nothing will be changed.\n");***  ***break;***  ***}***  ***if (init\_checkmark == 0) return 0;***  ***return 1;***  ***}***  ***int update\_port\_count(WIFI\* router, const unsigned short value)***  ***{***  ***router->port\_count = value;***  ***return router\_DataCheck(router);***  ***}***  ***int update\_vendor\_name(WIFI\* router, const char\* value)***  ***{***  ***strncpy(router->vendor, value, MAX\_VENDOR\_LENGTH);***  ***return router\_DataCheck(router);***  ***}***  ***int update\_5G\_mark(WIFI\* router, const enum \_5G\_MARK value)***  ***{***  ***router->has\_5G = value;***  ***return router\_DataCheck(router);***  ***}***  ***WIFI\* dynamic\_struct\_create(const char\* vendor, const unsigned short port\_count, const enum \_5G\_MARK has\_5G)***  ***{***  ***WIFI\* router = (WIFI\*)malloc(sizeof(WIFI));***    ***if (!router) return NULL;***  ***if (struct\_init\_result(init\_struct(router, vendor, port\_count, has\_5G)) == 0) return router;***  ***dynamic\_struct\_free(router);***  ***return NULL;***  ***}***  ***WIFI\* dynamic\_struct\_free(WIFI\* dym\_struct\_ptr)***  ***{***  ***free(dym\_struct\_ptr);***  ***dym\_struct\_ptr = NULL;***  ***}***  ***int is\_equal(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***return ((strcmp(router1->vendor, router2->vendor) == 0) && router1->port\_count == router2->port\_count && router1->has\_5G == router2->has\_5G);***  ***}***  ***int compare(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***int vend\_cmp\_result = strcmp(router2->vendor, router1->vendor);***  ***if (vend\_cmp\_result != 0) return (vend\_cmp\_result < 0);***  ***return compare\_int(router1, router2);***  ***}***  ***// Port\_count - Second priority of comparsion. 5G mark - third.***  ***// Returns 1 if p2 is bigger. Else 0.***  ***int compare\_int(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***if (router2->port\_count != router1->port\_count)***  ***{***  ***if ((router2->port\_count > router1->port\_count)) return 1;***  ***return 0;***  ***}***  ***if (router2->has\_5G > router1->has\_5G) return 1;***  ***return 0;***  ***}***  ***#endif*** |

***list.h***

|  |
| --- |
| ***#include "WIFI.h"***  ***#ifndef LIST\_H***  ***#define LIST\_H***  ***typedef struct WIFI\_item***  ***{***  ***WIFI data; // Данные о роутере по значению***  ***struct WIFI\_item\* next; // Указатель на следующий элемент списка***  ***}WIFI\_item;***  ***typedef struct WIFI\_list***  ***{***  ***struct WIFI\_item\* head; // Указатель на первый элемент списка***  ***}WIFI\_list;***  ***// Функция добавления элемента в начало списка***  ***// Принимает в качестве аргументов список роутеров по указателю, экземпляр структуры роутера по значению.***  ***// Выделяет память динамически для нового элемента.***  ***// Устанавливаются новые значения указателей next, head***  ***// Возвращает 0 при успешном исходе, иначе 1***  ***int push\_front(WIFI\_list\* list, WIFI data);***  ***// Функция удаления первого элемента списка***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// обновляет head, освобождает память для удаленного элемента.***  ***void delete\_front(WIFI\_list\* list);***  ***// Функция вывода сведений для каждого элемента списка***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// Внутри вызывает print\_struct***  ***void print\_list(const WIFI\_list\* list);***  ***// Функция очищения списка.***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// вызывает delete\_front, пока в списке есть элементы***  ***void delete\_list(WIFI\_list\* list);***  ***// Добавляет новый элемент в список с сохранением условий упорядочивания. Список упорядочен по 1)vendor 2)port\_count, has\_5G***  ***// Принимает в качестве аргументов список роутеров по указателю, экземпляр структуры роутера по значению.***  ***// При успешной инициализации нового элемента возвращает возвращает 0, иначе 1***  ***int add\_to\_ordered(WIFI\_list\* list, WIFI data);***  ***// Функция удаления заданного элемента из списка***  ***// Принимает в качестве аргументов список роутеров по указателю, экземпляр структуры роутера по значению.***  ***// Находит элемент через is\_equal***  ***// Если удаляемый элемент в начале списка обновляем head***  ***// Если нет, обновляем next. Если соответствующий элемент отсутствует в списке, выводится ошибка и список остаётся прежним.***  ***void delete\_selected\_element(WIFI\_list\* list, WIFI data);***  ***#endif*** |

***list.c***

|  |
| --- |
| ***#include "list.h"***  ***int push\_front(WIFI\_list\* list, WIFI data)***  ***{***  ***WIFI\_item\* new\_item = (WIFI\_item\*)malloc(sizeof(WIFI\_item));***  ***if (!new\_item)***  ***{***  ***printf("Can't create new item");***  ***return 1;***  ***}***  ***new\_item->data = data;***  ***new\_item->next = list->head;***  ***list->head = new\_item;***  ***return 0;***  ***}***  ***void delete\_front(WIFI\_list\* list)***  ***{***  ***if (list->head == NULL)***  ***{***  ***printf("Empty.\n");***  ***return;***  ***}***  ***struct WIFI\_item\* temp = list->head;***  ***list->head = list->head->next;***  ***free(temp);***  ***}***  ***void print\_list(const WIFI\_list\* list)***  ***{***  ***WIFI\_item\* current = list->head;***  ***while (current != NULL)***  ***{***  ***print\_struct(&current->data);***  ***current = current->next;***  ***}***  ***}***  ***void delete\_list(WIFI\_list\* list)***  ***{***  ***while (list->head != NULL) delete\_front(list);***  ***}***  ***int add\_to\_ordered(WIFI\_list\* list, WIFI data)***  ***{***  ***WIFI\_item\* new\_item = (WIFI\_item\*)malloc(sizeof(WIFI\_item));***  ***if (!new\_item)***  ***{***  ***printf("Can't create new item.");***  ***return 1;***  ***}***  ***new\_item->data = data;***  ***new\_item->next = NULL;***  ***if (list->head == NULL || compare(&data, &list->head->data) < 0)***  ***{***  ***new\_item->next = list->head;***  ***list->head = new\_item;***  ***}***  ***else***  ***{***  ***WIFI\_item\* current = list->head;***  ***while (current->next != NULL && compare(&data, &current->next->data) >= 0) current = current->next;***  ***new\_item->next = current->next;***  ***current->next = new\_item;***  ***}***  ***return 0;***  ***}***  ***void delete\_selected\_element(WIFI\_list\* list, WIFI data)***  ***{***  ***WIFI\_item\* current = list->head;***  ***WIFI\_item\* previous = NULL;***  ***while (current != NULL)***  ***{***  ***if (is\_equal(&current->data, &data))***  ***{***  ***if (previous == NULL) list->head = current->next;***  ***else previous->next = current->next;***  ***free(current);***  ***return;***  ***}***  ***previous = current;***  ***current = current->next;***  ***}***  ***printf("Selected element not in list.\n");***  ***}*** |

***main.c***

|  |
| --- |
| ***#include "WIFI.h"***  ***#include "list.h"***  ***int main()***  ***{***  ***// Инициализация трёх экземпляров структуры на стеке***  ***printf("Creating first struct.\n");***  ***WIFI router1;***  ***if (struct\_init\_result(init\_struct(&router1, "TPLINK", 2, Is5G)) != 0) return 1;***  ***printf("First: \n");***  ***print\_struct(&router1); // Вывод сведений о структуре***  ***printf("Creating second struct.\n");***  ***WIFI router2;***  ***if (struct\_init\_result(init\_struct(&router2, "TPLINK", 6, Is5G)) != 0) return 1;***  ***printf("Second: \n");***  ***print\_struct(&router2);***  ***printf("Creating third struct.\n");***  ***WIFI router3;***  ***if (struct\_init\_result(init\_struct(&router3, "ROSTELECOM", 2, Not5G)) != 0) return 1;***  ***printf("Third: \n");***  ***print\_struct(&router3);***  ***// Инициализация неупорядоченного списка***  ***printf("Initializing not ordered list with three structs.\n");***  ***WIFI\_list\* router\_list = (WIFI\_list\*)malloc(sizeof(WIFI\_list));***  ***if (router\_list == NULL)***  ***{***  ***printf("Can't create list.\n");***  ***return 1;***  ***}***  ***router\_list->head = NULL;***    ***// Добавляем элементы в начало списка***  ***if (push\_front(router\_list, router1) == 1) return 1;***  ***if (push\_front(router\_list, router2) == 1) return 1;***  ***if (push\_front(router\_list, router3) == 1) return 1;***  ***print\_list(router\_list); // Вывод сведений о каждом элементе списка***  ***printf("Removing front element:\n");***  ***delete\_front(router\_list); // Удаления первого элемента списка***    ***print\_list(router\_list);***  ***delete\_list(router\_list); // Удаление списка полностью.***    ***printf("Creating fourth struct.\n");***  ***WIFI router4;***  ***if (struct\_init\_result(init\_struct(&router4, "4R", 3, Is5G)) != 0) return 1;***  ***printf("Fourth: \n");***  ***print\_struct(&router1);***  ***printf("Initializing ordered list\n");***  ***if(add\_to\_ordered(router\_list, router1) == 1) return 1;***  ***if(add\_to\_ordered(router\_list, router2) == 1) return 1;***  ***if(add\_to\_ordered(router\_list, router3) == 1) return 1;***  ***if(add\_to\_ordered(router\_list, router4) == 1) return 1;***  ***print\_list(router\_list);***  ***delete\_selected\_element(router\_list, router4);***  ***printf("Ordered list after deleting element:\n");***  ***print\_list(router\_list);***  ***delete\_list(router\_list);***  ***return 0;***  ***}*** |

***После выполнения шага 10***

***WIFI.h***

|  |
| --- |
| ***#include <stddef.h>***  ***#include <stdio.h>***  ***#include <string.h>***  ***#include <stdlib.h>***  ***#include <limits.h>***  ***#include <math.h>***  ***#ifndef WIFI\_H***  ***#define WIFI\_H***  ***#define MAX\_VENDOR\_LENGTH 20***  ***#define MAX\_INT\_LENGTH 8***  ***#define MAX\_MARK\_LENGTH 10***  ***enum ERROR\_CODE***  ***{***  ***Success, Invalid\_Port\_Count, Invalid\_5G\_Mark***  ***};***  ***enum \_5G\_MARK***  ***{***  ***Is5G, Not5G, Undefined***  ***};***  ***typedef struct WIFI***  ***{***  ***char vendor[MAX\_VENDOR\_LENGTH]; // название бренда Wifi маршрутизатора***  ***unsigned short port\_count; // количество ethernet портов***  ***enum \_5G\_MARK has\_5G; // наличие диапазона Wi-Fi 5 ГГц***  ***}WIFI;***  ***int input\_int(const char\* message);***  ***void input\_string(char\* buffer, int buffer\_size);***  ***enum \_5G\_MARK input\_5G\_mark(const char\* message);***  ***enum ERROR\_CODE router\_DataCheck(const WIFI\* router);***  ***//Функция для инициализации структуры.***  ***//В качестве аргументов принимает структуру WIFI по указателю, символьный массив brand\_name, целочисленную переменную port\_count и переменную перечислимого типа has\_5G.***  ***int init\_struct(WIFI\* router, const char\* vendor, unsigned short port\_count, enum \_5G\_MARK has\_5G );***  ***int struct\_init\_result(const enum ERROR\_CODE result\_code);***  ***int init\_userdata(WIFI\* router);***  ***void copy\_struct(const WIFI\* router, WIFI\* copy);***  ***//Функция для вывода всех полей структуры.***  ***//В качестве аргумента получает структуру по указателю.***  ***void print\_struct(const WIFI\* router);***  ***int field\_changing(WIFI\* router);***  ***int update\_port\_count(WIFI\* router, const unsigned short value);***  ***int update\_vendor\_name(WIFI\* router, const char\* value);***  ***int update\_5G\_mark(WIFI\* router, const enum \_5G\_MARK value);***  ***WIFI\* dynamic\_struct\_create(const char\* vendor, const unsigned short port\_count, const enum \_5G\_MARK has\_5G);***  ***WIFI\* dynamic\_struct\_free(WIFI\* dym\_struct\_ptr);***  ***// Проверяет равенство двух экземпляров структуры посредством сравнения значений по каждому из полей экземпляров.***  ***// Возвращает 1, если экземпляры идентичны.***  ***int is\_equal(const WIFI\* router1, const WIFI\* router2);***  ***// Функция для сравнения двух экземпляров структур. Возвращает 0, если "больше" первый экземпляр, 1 если второй соответственно.***  ***// Приоритеты для сравнения определены: 1) vendor 2) port\_count, has\_5G.***  ***// Если vendor у экземпляров совпадает, производится вызов compare\_int, которая сравнивает экземпляры по второму приоритету.***  ***int compare(const WIFI\* router1, const WIFI\* router2);***  ***int compare\_int(const WIFI\* router1, const WIFI\* router2);***  ***#endif /\* WIFI\_H \*/*** |

***WIFI.c***

|  |
| --- |
| ***#define \_CRT\_SECURE\_NO\_WARNINGS***  ***#include "WIFI.h"***  ***#ifndef WIFI\_C***  ***#define WIFI\_C***  ***int input\_int(const char\* message)***  ***{***  ***char buffer[MAX\_INT\_LENGTH];***  ***char temp[MAX\_INT\_LENGTH];***  ***printf("%s", message);***  ***input\_string(buffer, MAX\_INT\_LENGTH);***  ***snprintf(temp, MAX\_INT\_LENGTH, "%d", atoi(buffer));***  ***if ((atoi(buffer) == 0 && buffer[0] != '0') || (atoi(buffer) >= USHRT\_MAX) || (atoi(buffer) < 0) || (strcmp(buffer, temp) != 0))***  ***{***  ***printf("Entered value is incorrect.");***  ***exit(EXIT\_FAILURE);***  ***}***  ***return atoi(buffer);***  ***}***  ***void input\_string(char\* buffer, int buffer\_size)***  ***{***  ***fgets(buffer, buffer\_size, stdin);***  ***size\_t len = strlen(buffer);***  ***if (len > 1 && buffer[len - 1] == '\n') buffer[len - 1] = '\0';***  ***else***  ***{***  ***printf("Entered value is too long or empty.\n");***  ***exit(EXIT\_FAILURE);***  ***}***  ***}***  ***enum \_5G\_MARK input\_5G\_mark(const char\* message)***  ***{***  ***char buffer[MAX\_MARK\_LENGTH];***  ***printf("%s", message);***  ***input\_string(buffer, MAX\_MARK\_LENGTH);***  ***if (strcmp(buffer, "yes") == 0) return Is5G;***  ***if (strcmp(buffer, "no") == 0) return Not5G;***  ***return Undefined;***  ***}***  ***enum ERROR\_CODE router\_DataCheck(const WIFI\* router)***  ***{***  ***if (router->port\_count > 32)***  ***{***  ***printf("Entered port count information is incorrect. Port count cant be more than 32.\n");***  ***return Invalid\_Port\_Count;***  ***}***  ***if(router->has\_5G == Undefined)***  ***{***  ***printf("Entered 5G mark is incorrect.");***  ***return Invalid\_5G\_Mark;***  ***}***  ***printf("Entered data is correct!\n");***  ***return Success;***  ***}***  ***int init\_struct(WIFI\* router, const char\* vendor, unsigned short port\_count, enum \_5G\_MARK has\_5G )***  ***{***  ***strncpy(router->vendor, vendor, MAX\_VENDOR\_LENGTH);***  ***router->vendor[MAX\_VENDOR\_LENGTH - 1] = '\0';***  ***router->port\_count = port\_count;***  ***router->has\_5G = has\_5G;***  ***return router\_DataCheck(router);***  ***}***  ***int struct\_init\_result(const enum ERROR\_CODE result\_code)***  ***{***  ***switch (result\_code)***  ***{***  ***case Success:***  ***printf("Structure initialized successfully.\n");***  ***return 0;***  ***default:***  ***printf("Structure initialization failed.\n");***  ***return 1;***  ***}***  ***}***  ***int init\_userdata(WIFI\* router)***  ***{***  ***char vendor[MAX\_VENDOR\_LENGTH];***  ***printf("Enter brand name: ");***  ***input\_string(vendor, MAX\_VENDOR\_LENGTH);***  ***unsigned short port\_count = input\_int("\nEnter port count: ");***  ***enum \_5G\_MARK has\_5G = input\_5G\_mark("\nHas 5G? (yes/no): ");***    ***if (struct\_init\_result(init\_struct(router, vendor, port\_count, has\_5G)) != 0) return 1;***  ***return 0;***  ***}***  ***void copy\_struct(const WIFI\* router, WIFI\* copy)***  ***{***  ***strcpy(copy->vendor, router->vendor);***  ***copy->port\_count = router->port\_count;***  ***copy->has\_5G = router->has\_5G;***  ***}***  ***void print\_struct(const WIFI\* router)***  ***{***  ***printf("Brand name: %s \nport\_count: %d\nhas 5g: %s\n\n ",***  ***router->vendor,***  ***router->port\_count,***  ***router->has\_5G ? "yes":"no");***  ***}***  ***int field\_changing(WIFI\* router)***  ***{***  ***int field\_tochange = input\_int("\nWhich field you want to change?\n1 - brand name\n2 - port count\n3 - Wifi 5G Mark\nEnter any other symbol if you won't change anything\n\nEnter only digit: ");***  ***short init\_checkmark;***  ***switch (field\_tochange)***  ***{***  ***case 1:***  ***char vendor\_name\_tochange[MAX\_VENDOR\_LENGTH];***  ***printf("\nEnter new vendor value: ");***  ***input\_string(vendor\_name\_tochange, MAX\_VENDOR\_LENGTH);***  ***init\_checkmark = struct\_init\_result(update\_vendor\_name(router, vendor\_name\_tochange));***  ***break;***  ***case 2:***  ***unsigned short port\_count\_tochange = input\_int("\nEnter new port count value: ");***  ***init\_checkmark = struct\_init\_result(update\_port\_count(router, port\_count\_tochange));***  ***break;***  ***case 3:***  ***unsigned short \_5G\_mark\_tochange = input\_5G\_mark("\nEnter new 5G mark value: ");***  ***init\_checkmark = struct\_init\_result(update\_5G\_mark(router, \_5G\_mark\_tochange));***  ***break;***  ***default:***  ***printf("\nUndefined value received. Nothing will be changed.\n");***  ***break;***  ***}***  ***if (init\_checkmark == 0) return 0;***  ***return 1;***  ***}***  ***int update\_port\_count(WIFI\* router, const unsigned short value)***  ***{***  ***router->port\_count = value;***  ***return router\_DataCheck(router);***  ***}***  ***int update\_vendor\_name(WIFI\* router, const char\* value)***  ***{***  ***strncpy(router->vendor, value, MAX\_VENDOR\_LENGTH);***  ***return router\_DataCheck(router);***  ***}***  ***int update\_5G\_mark(WIFI\* router, const enum \_5G\_MARK value)***  ***{***  ***router->has\_5G = value;***  ***return router\_DataCheck(router);***  ***}***  ***WIFI\* dynamic\_struct\_create(const char\* vendor, const unsigned short port\_count, const enum \_5G\_MARK has\_5G)***  ***{***  ***WIFI\* router = (WIFI\*)malloc(sizeof(WIFI));***    ***if (!router) return NULL;***  ***if (struct\_init\_result(init\_struct(router, vendor, port\_count, has\_5G)) == 0) return router;***  ***dynamic\_struct\_free(router);***  ***return NULL;***  ***}***  ***WIFI\* dynamic\_struct\_free(WIFI\* dym\_struct\_ptr)***  ***{***  ***free(dym\_struct\_ptr);***  ***dym\_struct\_ptr = NULL;***  ***}***  ***int is\_equal(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***return ((strcmp(router1->vendor, router2->vendor) == 0) && router1->port\_count == router2->port\_count && router1->has\_5G == router2->has\_5G);***  ***}***  ***int compare(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***int vend\_cmp\_result = strcmp(router2->vendor, router1->vendor);***  ***if (vend\_cmp\_result != 0) return (vend\_cmp\_result < 0);***  ***return compare\_int(router1, router2);***  ***}***  ***// Port\_count - Second priority of comparsion. 5G mark - third.***  ***// Returns 1 if p2 is bigger. Else 0.***  ***int compare\_int(const WIFI\* router1, const WIFI\* router2)***  ***{***  ***if (router2->port\_count != router1->port\_count)***  ***{***  ***if ((router2->port\_count > router1->port\_count)) return 1;***  ***return 0;***  ***}***  ***if (router2->has\_5G > router1->has\_5G) return 1;***  ***return 0;***  ***}***  ***#endif*** |

***list.h***

|  |
| --- |
| ***#include "WIFI.h"***  ***#ifndef LIST\_H***  ***#define LIST\_H***  ***typedef struct WIFI\_item***  ***{***  ***WIFI data; // Данные о роутере по значению***  ***struct WIFI\_item\* next; // Указатель на следующий элемент списка***  ***}WIFI\_item;***  ***typedef struct WIFI\_list***  ***{***  ***struct WIFI\_item\* head; // Указатель на первый элемент списка***  ***}WIFI\_list;***  ***// Функция добавления элемента в начало списка***  ***// Принимает в качестве аргументов список роутеров по указателю, экземпляр структуры роутера по значению.***  ***// Выделяет память динамически для нового элемента.***  ***// Устанавливаются новые значения указателей next, head***  ***// Возвращает 0 при успешном исходе, иначе 1***  ***int push\_front(WIFI\_list\* list, WIFI data);***  ***// Функция удаления первого элемента списка***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// обновляет head, освобождает память для удаленного элемента.***  ***void delete\_front(WIFI\_list\* list);***  ***// Функция вывода сведений для каждого элемента списка***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// Внутри вызывает print\_struct***  ***void print\_list(const WIFI\_list\* list);***  ***// Функция очищения списка.***  ***// Принимает в качестве аргумента список роутеров по указателю.***  ***// вызывает delete\_front, пока в списке есть элементы***  ***void delete\_list(WIFI\_list\* list);***  ***#endif*** |

***list.c***

|  |
| --- |
| ***#include "list.h"***  ***int push\_front(WIFI\_list\* list, WIFI data)***  ***{***  ***WIFI\_item\* new\_item = (WIFI\_item\*)malloc(sizeof(WIFI\_item));***  ***if (!new\_item)***  ***{***  ***printf("Can't create new item");***  ***return 1;***  ***}***  ***new\_item->data = data;***  ***new\_item->next = list->head;***  ***list->head = new\_item;***  ***return 0;***  ***}***  ***void delete\_front(WIFI\_list\* list)***  ***{***  ***if (list->head == NULL)***  ***{***  ***printf("Empty.\n");***  ***return;***  ***}***  ***struct WIFI\_item\* temp = list->head;***  ***list->head = list->head->next;***  ***free(temp);***  ***}***  ***void print\_list(const WIFI\_list\* list)***  ***{***  ***WIFI\_item\* current = list->head;***  ***while (current != NULL)***  ***{***  ***print\_struct(&current->data);***  ***current = current->next;***  ***}***  ***}***  ***void delete\_list(WIFI\_list\* list)***  ***{***  ***while (list->head != NULL) delete\_front(list);***  ***}*** |

***main.c***

|  |
| --- |
| ***#include "WIFI.h"***  ***#include "list.h"***  ***int main()***  ***{***  ***// Инициализация трёх экземпляров структуры на стеке***  ***printf("Creating first struct.\n");***  ***WIFI router1;***  ***if (struct\_init\_result(init\_struct(&router1, "TPLINK", 2, Is5G)) != 0) return 1;***  ***printf("First: \n");***  ***print\_struct(&router1); // Вывод сведений о структуре***  ***printf("Creating second struct.\n");***  ***WIFI router2;***  ***if (struct\_init\_result(init\_struct(&router2, "TPLINK", 6, Is5G)) != 0) return 1;***  ***printf("Second: \n");***  ***print\_struct(&router2);***  ***printf("Creating third struct.\n");***  ***WIFI router3;***  ***if (struct\_init\_result(init\_struct(&router3, "ROSTELECOM", 2, Not5G)) != 0) return 1;***  ***printf("Third: \n");***  ***print\_struct(&router3);***  ***// Инициализация неупорядоченного списка***  ***printf("Initializing not ordered list with three structs.\n");***  ***WIFI\_list\* router\_list = (WIFI\_list\*)malloc(sizeof(WIFI\_list));***  ***if (router\_list == NULL)***  ***{***  ***printf("Can't create list.\n");***  ***return 1;***  ***}***  ***router\_list->head = NULL;***    ***// Добавляем элементы в начало списка***  ***if (push\_front(router\_list, router1) == 1) return 1;***  ***if (push\_front(router\_list, router2) == 1) return 1;***  ***if (push\_front(router\_list, router3) == 1) return 1;***  ***print\_list(router\_list); // Вывод сведений о каждом элементе списка***  ***printf("Removing front element:\n");***  ***delete\_front(router\_list); // Удаления первого элемента списка***    ***print\_list(router\_list);***  ***delete\_list(router\_list); // Удаление списка полностью.***  ***return 0;***  ***}*** |