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: Файлы (небуферизированный ввод-вывод)

|  |  |
| --- | --- |
| Выполнили:  Кривощапов Сергей,  Пирожников Владислав,  Писарев Радомир  Группа АВТ-414  Факультет АВТФ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «5» мая 2025 г. | Проверил:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ф. И. О.)  Балл: \_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

Новосибирск 2025

[1. Проверить работу функций open и close. 3](#_Toc1)

[1.1. 3](#_Toc2)

[1.2. 4](#_Toc3)

[1.3 5](#_Toc4)

[2. Проверить работу функции write. 6](#_Toc5)

[2.1 6](#_Toc6)

[2.2 7](#_Toc7)

[2.3 8](#_Toc8)

[3. Проверить работу функции read. 9](#_Toc9)

[3.1 9](#_Toc10)

[3.2 10](#_Toc11)

[4. Проверить работу функции lseek. 12](#_Toc12)

[4.1 12](#_Toc13)

[4.2 13](#_Toc14)

[4.3 14](#_Toc15)

[5. Работа со структурами в файле. 15](#_Toc16)

[5.1 15](#_Toc17)

[WIFI.c 15](#_Toc18)

[WIFI.h 20](#_Toc19)

[main.c 21](#_Toc20)

[5.2 22](#_Toc21)

[WIFI.c 22](#_Toc22)

[WIFI.h 27](#_Toc23)

[main.c 29](#_Toc24)

[5.3 31](#_Toc25)

[WIFI.c 31](#_Toc26)

[WIFI.h 35](#_Toc27)

[my\_queue.h 37](#_Toc28)

[my\_queue.c 39](#_Toc29)

[main.c 42](#_Toc30)

[Вывод 45](#_Toc31)

# 1. Проверить работу функций open и close.

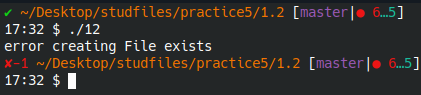
## 1.1.

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  int main()  {  int fd = open("1.bin", O\_CREAT | O\_WRONLY, 0644);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file opened\n");  if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("file closed\n");  return 0;  } |

## 1.2.

|  |
| --- |
| #include <stdio.h>  #include <fcntl.h>  #include <errno.h>  #include <unistd.h>  #include <string.h>  int main()  {  int fd = open("12.bin", O\_WRONLY | O\_CREAT | O\_EXCL, 0644);  if (fd == -1)  {  printf("error creating %s\n", strerror(errno));  return 1;  }  printf("file created and opened\n");    if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("file closed\n");  fd = open("12.bin", O\_WRONLY | O\_CREAT | O\_EXCL, 0644);  if (fd == -1)  {  printf("error creating !! %s\n", strerror(errno));  return 1;  }  printf("file created and opened\n");    if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("file closed\n");    return 0;  } |

Скриншот сообщения об ошибке



## 1.3

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  int main()  {  int fd = open("1.bin", O\_CREAT | O\_WRONLY | O\_APPEND, 0644);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file opened. is able to append in it.\n");  if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("file closed\n");  return 0;  } |

Отличия от программы из п. 1.1:

В программе 1.3 гарантируется, что файл открыт для дозаписи. Все записи в нем будут производиться в конец файла. В 1.1 файл может быть перезаписан, если не установить указатель вручную.

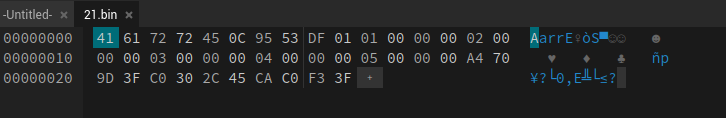
# 2. Проверить работу функции write.

## 2.1

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  int main()  {  int fd = open("21.bin", O\_CREAT | O\_WRONLY | O\_TRUNC, 0644);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file created and opened.\n");    char c = 'A';  write(fd, &c, sizeof(c));  char arr[] = "arr";  write(fd, arr, strlen(arr));  short s = 3141;  write(fd, &s, sizeof(s));  int i = 31413141;  write(fd, &i, sizeof(i));  int i\_arr[] = {1, 2, 3, 4, 5};  write(fd, i\_arr, sizeof(i\_arr));  float f = 1.23f;  write(fd, &f, sizeof(f));  double d = 1.2345678999;  write(fd, &d, sizeof(d));  if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("data written. file closed.\n");  return 0;  } |

## 2.2

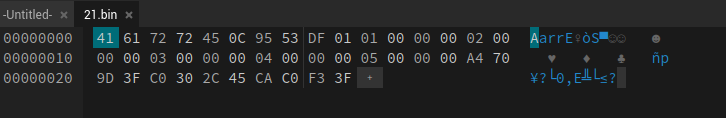
Бинарное представление данных в шестнадцатеричном формате:



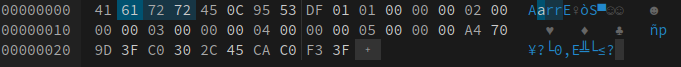
## 2.3

Рассматриваемые байты веделены голубым цветом

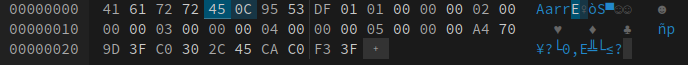
1)‘A’ (char)



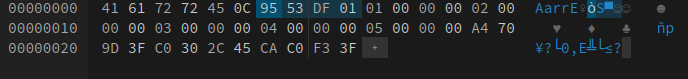
2) “arr” (массив char)



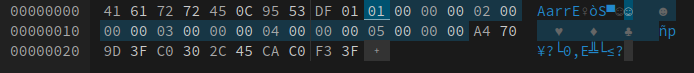
3) 3141 (short)



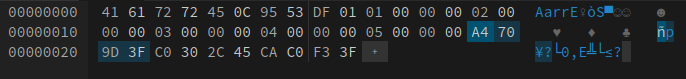
4) 31413141 (int)



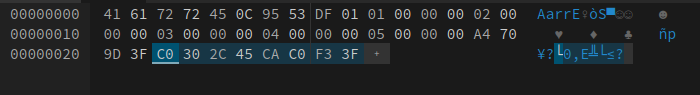
5) {1, 2, 3, 4, 5} (массив int)



6) 1.23 (float)



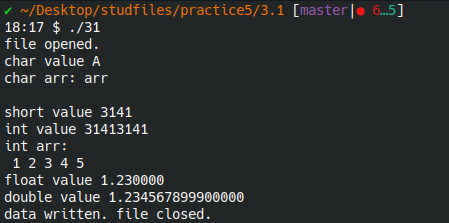
7) 1.2345678999 (double)



# 3. Проверить работу функции read.

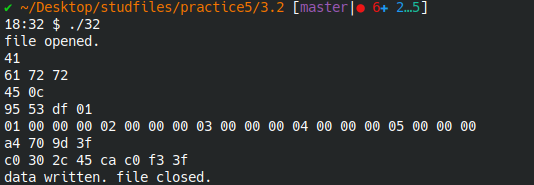
## 3.1

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  #define C 3  #define N 5  int main()  {  int fd = open("21.bin", O\_RDONLY);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file opened.\n");    char c;  read(fd, &c, sizeof(c));  printf("char value %c\n", c);    char arr[C+1];  read(fd, arr, C);  arr[C] = '\n';  printf("char arr: %s\n", arr);  short s;  read(fd, &s, sizeof(s));  printf("short value %hd\n", s);  int i;  read(fd, &i, sizeof(i));  printf("int value %d\n", i);  int i\_arr[N];  read(fd, i\_arr, sizeof(i\_arr));  printf("int arr:\n");  for (int count = 0; count < N; count++) printf(" %d", i\_arr[count]);  float f;  read(fd, &f, sizeof(f));  printf("\nfloat value %f\n", f);  double d;  read(fd, &d, sizeof(d));  printf("double value %.15lf\n", d);  if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("data written. file closed.\n");  return 0;  } |



## 3.2

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  #define N 5  #define C 3  void print\_bytes(void \*ptr, int size)  {  unsigned char \*p = ptr;  for (int i = 0; i < size; i++) printf("%02x ", p[i]);  printf("\n");  }  int main()  {  int fd = open("21.bin", O\_RDONLY);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file opened.\n");    char c;  read(fd, &c, sizeof(c));  print\_bytes(&c, sizeof(c));    char arr[C+1];  read(fd, arr, C);  print\_bytes(arr, C);  short s;  read(fd, &s, sizeof(s));  print\_bytes(&s, sizeof(s));  int i;  read(fd, &i, sizeof(i));  print\_bytes(&i, sizeof(i));  int i\_arr[N];  read(fd, i\_arr, sizeof(i\_arr));  print\_bytes(i\_arr, sizeof(i\_arr));  float f;  read(fd, &f, sizeof(f));  print\_bytes(&f, sizeof(f));  double d;  read(fd, &d, sizeof(d));  print\_bytes(&d, sizeof(d));  if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("data written. file closed.\n");  return 0;  } |



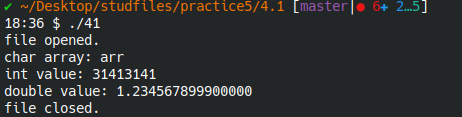
В отличии от результатов работы программы 3.1, результат работы 3.2 – побайтовое представление считанных значений. А 3.1 выводит считанные значения в формате, читаемом для человека.

Результат работы программы 3.2 полностью соответствует бинарному представлению в шестнадцатеричном формате из пункта 2.3.

# 4. Проверить работу функции lseek.

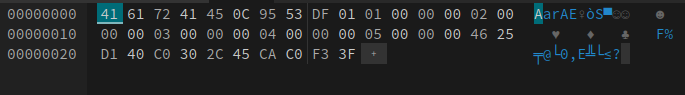
## 4.1

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  #define N 5  #define C 3  int main()  {  int fd = open("21.bin", O\_RDONLY);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file opened.\n");    lseek(fd, 1, SEEK\_SET); // char  char arr[C+1];  read(fd, arr, C);  printf("char array: %s\n", arr);  lseek(fd, 2, SEEK\_CUR); // short  int i;  read(fd, &i, sizeof(i));  printf("int value: %d\n", i);  lseek(fd, N \* (int)sizeof(int) + sizeof(float), SEEK\_CUR); // int array, float  double d;  read(fd, &d, sizeof(d));  printf("double value: %.15lf\n", d);  if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("file closed.\n");  return 0;  } |



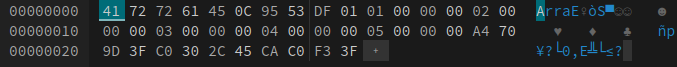
## 4.2

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  #define N 5  #define C 3  int main()  {  int fd = open("21.bin", O\_RDWR);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file opened.\n");    lseek(fd, (int)sizeof(char) \* (1 + 2) , SEEK\_SET); // char + 0, 1 index of array  char char\_tochange = 'A';  write(fd, &char\_tochange, sizeof(char\_tochange));  lseek(fd, sizeof(short) + sizeof(int) + sizeof(int) \* 5 , SEEK\_CUR); // short, int, int array    float float\_tochange = 6.5358f;  write(fd, &float\_tochange, sizeof(float\_tochange));  if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("file closed.\n");  return 0;  } |



## 4.3

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #include <string.h>  #define N 5  #define C 3  int main()  {  int fd = open("21.bin", O\_RDWR);  if (fd == -1)  {  printf("error opening %s\n", strerror(errno));  return 1;  }  printf("file opened.\n");  lseek(fd, 1, SEEK\_SET); // char  char arr[C];  read(fd, arr, C);  int arr\_half = C / 2;  for (int i = 0; i < arr\_half; i++)  {  char tmp = arr[i];  arr[i] = arr[C - i - 1];  arr[C - i - 1] = tmp;  }  lseek(fd, 1, SEEK\_SET);  write(fd, arr, C);    if (close(fd) == -1)  {  printf("error closing %s\n", strerror(errno));  return 1;  }  printf("file closed.\n");  return 0;  } |



Изменения корректны.

# 5. Работа со структурами в файле.

## 5.1

### WIFI.c

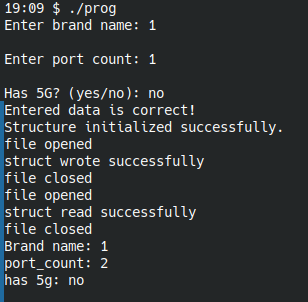
|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include "WIFI.h"  #ifndef WIFI\_C  #define WIFI\_C  int input\_int(const char\* message)  {  char buffer[MAX\_INT\_LENGTH];  char temp[MAX\_INT\_LENGTH];  printf("%s", message);  input\_string(buffer, MAX\_INT\_LENGTH);  snprintf(temp, MAX\_INT\_LENGTH, "%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;  }  // return 1 = error  int write\_tofile(const char\* filename, const WIFI\* router)  {  int fd = open(filename, O\_WRONLY | O\_CREAT | O\_APPEND, 0644);  if (fd == -1)  {  printf("error opening file %s\n", strerror(errno));  return 1;  }  printf("file opened\n");  write(fd, router, sizeof(WIFI));  printf("struct wrote successfully\n");  if (close(fd) == -1)  {  printf("error closing file %s\n", strerror(errno));  return 1;  }  printf("file closed\n");  return 0;  }  // return 1 = error  int read\_fromfile(const char\* filename, WIFI\* router)  {  int fd = open(filename, O\_RDONLY);  if (fd == -1)  {  printf("error opening file %s\n", strerror(errno));  return 1;  }  printf("file opened\n");  read(fd, router, sizeof(WIFI));  printf("struct read successfully\n");  if (close(fd) == -1)  {  printf("error closing file %s\n", strerror(errno));  return 1;  }    printf("file closed\n");  return 0;  }  #endif |

### WIFI.h

|  |
| --- |
| #include <stddef.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <limits.h>  #include <math.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.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);  int write\_tofile(const char\* filename, const WIFI\* router);  int read\_fromfile(const char\* filename, WIFI\* router);  #endif /\* WIFI\_H \*/ |

### main.c

|  |
| --- |
| #include "WIFI.h"  int main()  {  const char\* filename = "file.bin";  WIFI router1;    if (init\_userdata(&router1) == 1) return 1;  if (write\_tofile(filename, &router1) == 1) return 1;  WIFI router2;  if (read\_fromfile(filename, &router2) == 1) return 1;  print\_struct(&router2);  return 0;  } |



## 5.2

### WIFI.c

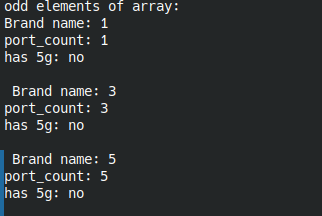
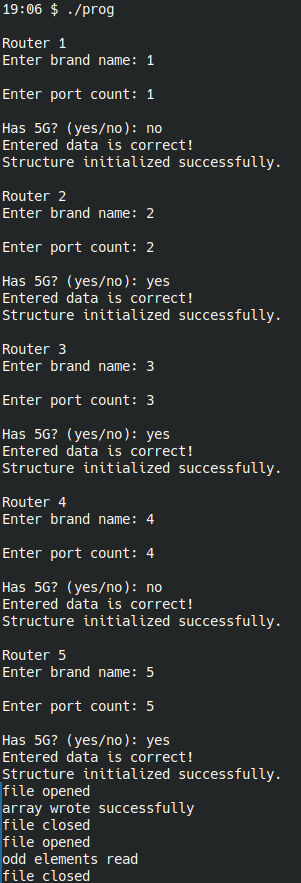
|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include "WIFI.h"  #ifndef WIFI\_C  #define WIFI\_C  int input\_int(const char\* message)  {  char buffer[MAX\_INT\_LENGTH];  char temp[MAX\_INT\_LENGTH];  printf("%s", message);  input\_string(buffer, MAX\_INT\_LENGTH);  snprintf(temp, MAX\_INT\_LENGTH, "%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;  }  // return 1 = error  int write\_arraytofile(const char\* filename, const WIFI\* routers, int array\_size)  {  int fd = open(filename, O\_WRONLY | O\_CREAT | O\_APPEND, 0644);  if (fd == -1)  {  printf("error opening file %s\n", strerror(errno));  return 1;  }  printf("file opened\n");  write(fd, routers, sizeof(WIFI) \* array\_size);  printf("array wrote successfully\n");  if (close(fd) == -1)  {  printf("error closing file %s\n", strerror(errno));  return 1;  }  printf("file closed\n");  return 0;  }  // return 1 = error  int read\_oddfromfile(const char\* filename, WIFI\* odd\_routers, int array\_size)  {  int fd = open(filename, O\_RDONLY);  if (fd == -1)  {  printf("error opening file %s\n", strerror(errno));  return -1;  }  printf("file opened\n");  int odd\_counter = 0;    for (int i = 0; i < array\_size; i += 2)  {  lseek(fd, i \* sizeof(WIFI), SEEK\_SET);  read(fd, &odd\_routers[odd\_counter], sizeof(WIFI));  odd\_counter++;  }    printf("odd elements read\n");  if (close(fd) == -1)  {  printf("error closing file %s\n", strerror(errno));  return -1;  }    printf("file closed\n");  return odd\_counter;  }  #endif |

### WIFI.h

|  |
| --- |
| #include <stddef.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <limits.h>  #include <math.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.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);  int write\_arraytofile(const char\* filename, const WIFI\* routers, int array\_size);  int read\_oddfromfile(const char\* filename, WIFI\* router, int array\_size);  #endif /\* WIFI\_H \*/ |

### main.c

|  |
| --- |
| #include "WIFI.h"  #define ARRAY\_SIZE 5  int main()  {  const char\* filename = "file.bin";  WIFI routers[ARRAY\_SIZE];  WIFI odd\_routers[ARRAY\_SIZE / 2 + 1];  for (int i = 0; i < ARRAY\_SIZE; i++)  {  printf("\nRouter %d\n", i + 1);  if (init\_userdata(&routers[i]) == 1) return 1;  }  if (write\_arraytofile(filename, routers, ARRAY\_SIZE) == 1) return 1;    int odd\_count = read\_oddfromfile(filename, odd\_routers, ARRAY\_SIZE);  if (odd\_count < 0) return 1;  printf("odd elements of array:\n");  for (int i = 0; i < odd\_count; i++) print\_struct(&odd\_routers[i]);  return 0;  } |



Отличия от программы 1.1:

Программа 1.1 создаёт и открывает файл только для записи и закрывает файл.

Программа 5.2 использует файлы для обработки массива структур: записывает в файл весь массив структур, считывает из него только нечетные элементы массива структур. Файл создается, если до этого не существовал, если существовал, до его длина сокращается до нуля.

## 5.3

### 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;  }  // return 1 = error  int write\_arraytofile(const char\* filename, const WIFI\* routers, int array\_size)  {  int fd = open(filename, O\_WRONLY | O\_CREAT | O\_TRUNC, 0644);  if (fd == -1)  {  printf("error opening file %s\n", strerror(errno));  return 1;  }  printf("file opened\n");  write(fd, routers, sizeof(WIFI) \* array\_size);  printf("array wrote successfully\n");  if (close(fd) == -1)  {  printf("error closing file %s\n", strerror(errno));  return 1;  }  printf("file closed\n");  return 0;  }  #endif |

### WIFI.h

|  |
| --- |
| #include <stddef.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <limits.h>  #include <math.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.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);  int write\_arraytofile(const char\* filename, const WIFI\* routers, int array\_size);  #endif /\* WIFI\_H \*/ |

### my\_queue.h

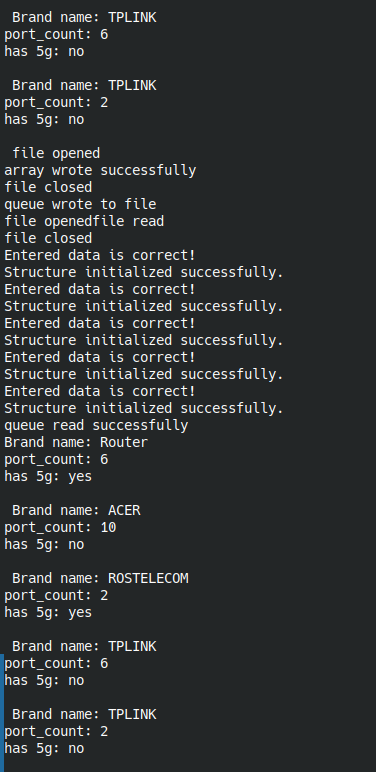
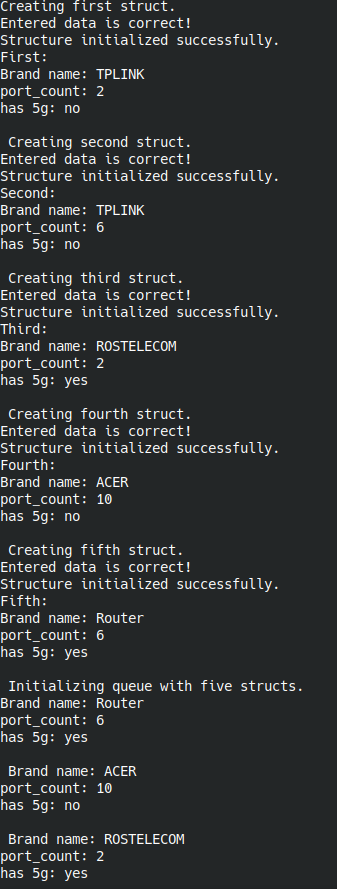
|  |
| --- |
| #include <stddef.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <limits.h>  #include <math.h>  #include <errno.h>  #include <unistd.h>  #include <fcntl.h>  #ifndef WIFI\_H  #define WIFI\_H  #define MAX\_VENDOR\_LENGTH 20  #define MAX\_INT\_LENGTH 8  #define 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);  int write\_arraytofile(const char\* filename, const WIFI\* routers, int array\_size);  #endif /\* WIFI\_H \*/ |

### my\_queue.c

|  |
| --- |
| #include "my\_queue.h"  int enqueue(WIFI\_queue\* queue, 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;  if (queue->head == NULL)  {  new\_item->next = queue->head;  queue->head = new\_item;  new\_item->prev = NULL;  return 0;  }  new\_item->next = queue->head;  queue->head->prev = new\_item;  queue->head = new\_item;  return 0;  }  WIFI\* dequeue(WIFI\_queue\* queue)  {  WIFI\* data = NULL;  if (queue->head == NULL)  {  printf("Empty.\n");  return data;  }    struct WIFI\_item\* temp = queue->head;  while (temp->next) temp = temp->next;    data = temp->data;    if (temp->prev) temp->prev->next = NULL;  else queue->head = NULL;    free(temp);  temp = NULL;    return data;  }  void extract\_data(WIFI\_queue\* queue)  {  WIFI\* data = dequeue(queue);  if (data != NULL)  {  printf("Extracted from queue: \n");  print\_struct(data);  }  }  void print\_queue(const WIFI\_queue\* queue)  {  WIFI\_item\* current = queue->head;  while (current != NULL)  {  print\_struct(current->data);  current = current->next;  }  }  void delete\_queue(WIFI\_queue\* queue)  {  while (queue->head != NULL) dequeue(queue);  }  // error = return -1  int write\_queue\_tofile(const char\* filename, WIFI\_queue\* queue)  {  int count = 0;  WIFI\_item\* current = queue->head;  while (current)  {  count++;  current = current->next;  }  WIFI\* routers = malloc(sizeof(WIFI) \* count);  if (!routers)  {  printf("error to allocate\n");  return -1;  }  current = queue->head;  for (int i = 0; i < count; i++)  {  copy\_struct(current->data, &routers[i]);  current = current->next;  }  int wrote = write\_arraytofile(filename, routers, count);  free(routers);  return wrote;  }  // error = return 1  int read\_queue\_fromfile(const char\* filename, WIFI\_queue\* queue)  {  int fd = open(filename, O\_RDONLY);  if (fd == - 1)  {  printf("error opening file\n");  return 1;  }  printf("file opened");  int size = lseek(fd, 0, SEEK\_END);  int count = size / sizeof(WIFI);  lseek(fd, 0, SEEK\_SET);  WIFI\* routers = malloc(sizeof(WIFI) \* count);  if (!routers)  {  printf("error to allocate\n");  return 1;  }  if (read(fd, routers, size) == -1)  {  printf("error reading file\n");  return 1;  }  printf("file read\n");  if (close(fd) == -1)  {  printf("error closing\n");  }  printf("file closed\n");  for (int i = count - 1; i >= 0; i--)  {  WIFI\* router = dynamic\_struct\_create(routers[i].vendor, routers[i].port\_count, routers[i].has\_5G);  if (!router || enqueue(queue, router) == 1)  {  printf("error adding to queue\n");  free(routers);  delete\_queue(queue);  return 1;  }  }  free(routers);  return 0;  } |

### main.c

|  |
| --- |
| #include "WIFI.h"  #include "my\_queue.h"  #define ARRAY\_SIZE 5  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);    printf("Creating fourth struct.\n");  WIFI\* router4 = dynamic\_struct\_create("ACER", 10, Is5G);    if (router4 == NULL) return 1;  printf("Fourth: \n");  print\_struct(router4);  printf("Creating fifth struct.\n");  WIFI\* router5 = dynamic\_struct\_create("Router", 6, Not5G);    if (router5 == NULL) return 1;  printf("Fifth: \n");  print\_struct(router5);  // Инициализация очереди  printf("Initializing queue with five structs.\n");  WIFI\_queue\* router\_queue = (WIFI\_queue\*)malloc(sizeof(WIFI\_queue));  if (router\_queue == NULL)  {  printf("Can't create queue.\n");  return 1;  }  router\_queue->head = NULL;  // Добавляем элементы в начало очереди  if (enqueue(router\_queue, router1) == 1) return 1;  if (enqueue(router\_queue, router2) == 1) return 1;  if (enqueue(router\_queue, router3) == 1) return 1;  if (enqueue(router\_queue, router4) == 1) return 1;  if (enqueue(router\_queue, router5) == 1) return 1;    print\_queue(router\_queue);  const char\* filename = "file.bin";  if (write\_queue\_tofile(filename, router\_queue) == -1) {  printf("Error saving queue to file\n");  delete\_queue(router\_queue);  return 1;  }  printf("queue wrote to file\n");    delete\_queue(router\_queue);  WIFI\_queue\* router\_queue2 = (WIFI\_queue\*)malloc(sizeof(WIFI\_queue));  if (router\_queue2 == NULL)  {  printf("Can't create queue.\n");  return 1;  }  router\_queue2->head = NULL;  if (read\_queue\_fromfile(filename, router\_queue2) == 1)  {  printf("error reading queue from file\n");  return 1;  }  printf("queue read successfully\n");  print\_queue(router\_queue2);  delete\_queue(router\_queue2);  dynamic\_struct\_free(router1);  dynamic\_struct\_free(router2);  dynamic\_struct\_free(router3);  dynamic\_struct\_free(router4);  dynamic\_struct\_free(router5);  return 0;  } |



# Вывод

Была освоена работа с небуферизированным файловым вводом/выводом. Изучена работа с функциями open(), close(), write(), read(), lseek(). Рассмотрено применение этих функций для взаимодействия с различными типами данных, структурами, массивами структур, очередями.