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: Структуры, функции, динамическая память. Вариант 11.

|  |  |
| --- | --- |
| Выполнили:  Кривощапов Сергей Владиславович  (Ф. И. О.)  Группа АВТ-414  Факультет АВТФ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «10» марта 2025 г. | Проверил:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ф. И. О.)  Балл: \_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

Новосибирск 2025

**Ответы на вопросы по шагам 1 – 3.**

1)

Имя структуры: discipline.

Поля структуры:

subject\_name — название учебного предмета.

total\_hours — общее количество часов.

lectures\_hours — количество часов лекций.

practice\_hours — количество часов практики.

examination\_form — форма аттестации.

2)

subject\_name — массив символов (char []).

total\_hours, lectures\_hours, practice\_hours — неотрицательные целые числа (unsigned short).

examination\_form — перечислимый тип (enum attestation\_form).

3)

subject\_name — символьный массив длиной до 24 символов.

total\_hours, lectures\_hours, practice\_hours — целые числа от 0 до 65 535 (диапазон значений unsigned short).

examination\_form — одно из значений: Zachet, Exam, Undefined.

total\_hours должно быть больше или равно сумме lectures\_hours и practice\_hours.

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

|  |
| --- |
| struct discipline  {  char subject\_name[MAX\_SUBJ\_NAME\_LENGTH]; // Название учебного предмета  unsigned short total\_hours; // Общее количество часов  unsigned short lectures\_hours; // Количество часов лекций  unsigned short practice\_hours; // Количество часов практики  enum attestation\_form examination\_form; // Форма аттестации  }; |

**Прототипы разработанных функций. Комментарии об особенностях их работы (алгоритм).**

|  |
| --- |
| //Проверяет, что общее количество часов >= сумме часов лекций и практик, что examination\_form не равен Undefined.  //Возвращает Success, если данные корректны, иначе возвращает код ошибки.  enum error\_code subject\_DataCheck(const struct discipline\* subj); |

|  |
| --- |
| //Инициализирует поля структуры переданными значениями.  //Вызывает subject\_DataCheck для проверки корректности данных.  enum error\_code subject\_init(struct discipline\* subj, const char\* subject\_name,  unsigned short toral\_hours, unsigned short lectural\_hours,  unsigned short practice\_hours, enum attestation\_form examination\_form); |

|  |
| --- |
| //Копирует значения всех полей экземпляра структуры src в поля экземпляра структуры dest.  void copy\_subject(struct discipline\* dest, const struct discipline\* src); |

|  |
| --- |
| //Выводит значения каждого из полей экземпляра структуры.  void subject\_print(struct discipline\* subj); |

|  |
| --- |
| //Получает на ввод из входного потока заранее объявленный номер изменяемого поля.  //Если полученное значение некорректно, не производит дальнейших действий.  //Получает на ввод значение для выбранного поля.  //Изменяет соответствующее значение поля экземпляра структуры посредством вызова одной из соответствующих функций.  //Проверяет корректность изменений, вызывая функцию struct\_init\_result.  //Если изменения корректны, возвращает 0, если некорректны 1.  int field\_changing(struct discipline\* subj); |

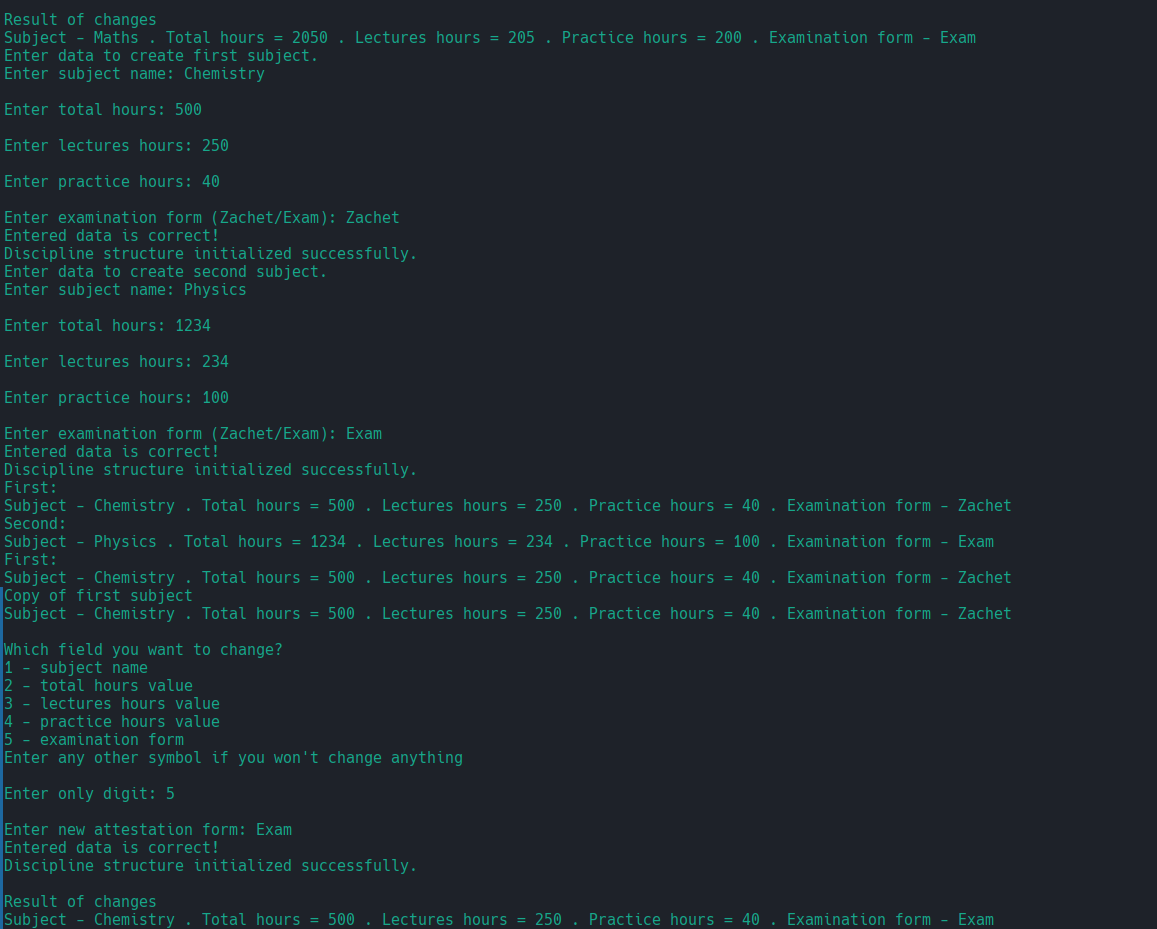
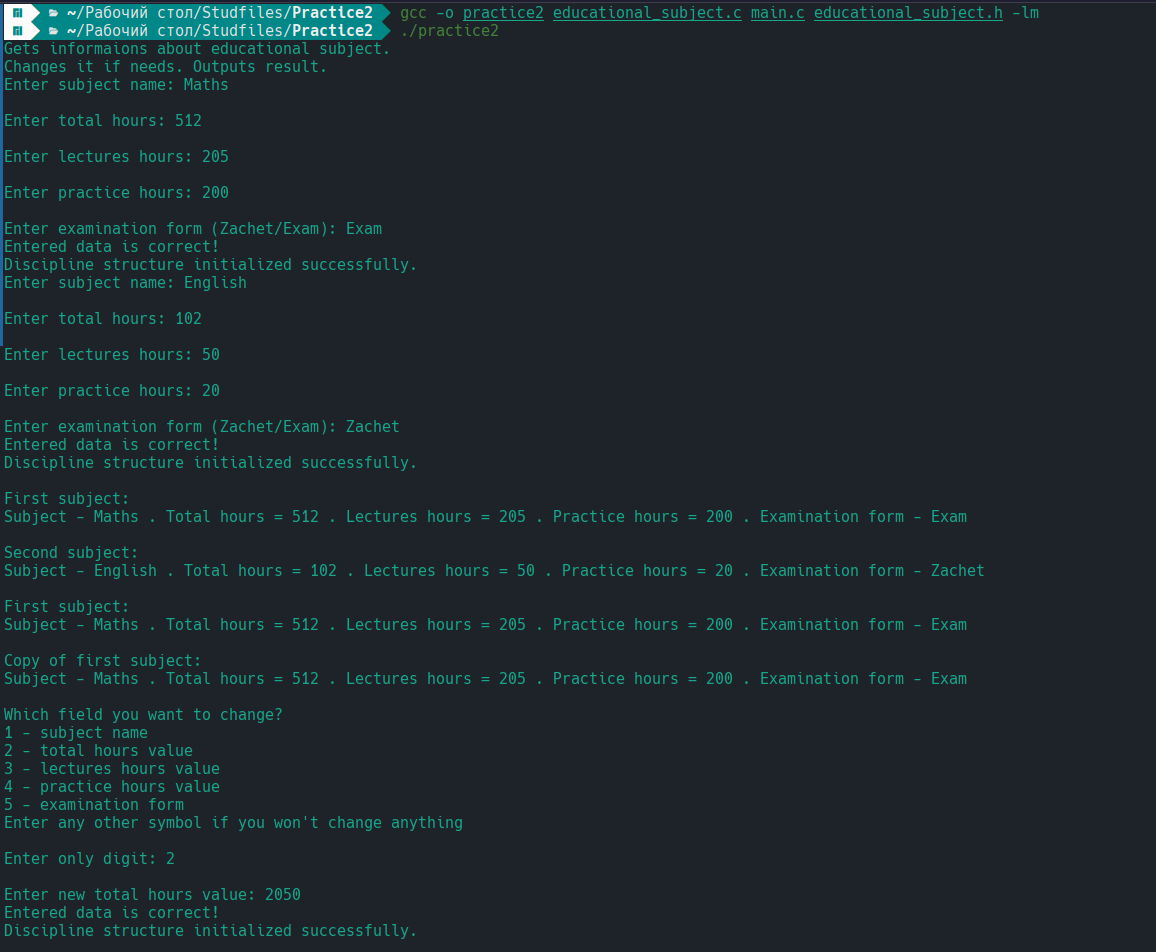
|  |
| --- |
| //Выделяет память для экземпляра структуры динамически.  //Инициализирует поля экземпляра значениями, полученными от пользователя через init\_userdata.  struct discipline\* create\_subject(); |

|  |
| --- |
| //Освобождает память, выделенную для экземпляра структуры.  //Присваивает значение NULL указателю на освобожденную память.  void delete\_subject(struct discipline \*subj); |

**Текст тестирующей функции с комментариями.**

|  |
| --- |
| #include "educational\_subject.h"  int main(int argc, char\* argv[])  {  // Создание экземпляров структур на стеке  struct discipline subj1;  struct discipline subj2;    printf("Gets informaions about educational subject.\n");  printf("Changes it if needs. Outputs result.\n");    // Инициализация экземпляров структуры  if (init\_userdata(&subj1) == 1) return 1;    if (init\_userdata(&subj2) == 1) return 1;    // Вывод сведений о экземплярах структуры  printf("\nFirst subject:\n");  subject\_print(&subj1);  printf("\nSecond subject:\n");  subject\_print(&subj2);  // Инициализация второго экземпляра структуры как копии первого  copy\_subject(&subj2, &subj1);  printf("\nFirst subject:\n");  subject\_print(&subj1);  printf("\nCopy of first subject:\n");  subject\_print(&subj2);  // Вызов функции изменения поля первого экземпляра структуры  if (field\_changing(&subj1) == 1) return 1;  printf("\nResult of changes\n");  subject\_print(&subj1);    // Инициализация двух экземпляров структур в динамической памяти  printf("Enter data to create first subject.\n");  struct discipline\* subj3 = create\_subject();    if (subj3 == NULL) return 1;    printf("Enter data to create second subject.\n");  struct discipline\* subj4 = create\_subject();  if (subj4 == NULL) return 1;  printf("First: \n");  subject\_print(subj3);  printf("Second: \n");  subject\_print(subj4);    // Инициализируем второй экземпляр структуры как копию первого  copy\_subject(subj4, subj3);    printf("First: \n");  subject\_print(subj3);  printf("Copy of first subject\n");  subject\_print(subj4);  // Вызов функции изменения поля в экземпляре структуры, инициализированном в динамической памяти  if (field\_changing(subj3) == 1) return 1;    printf("\nResult of changes\n");  subject\_print(subj3);  // Удаление экземпляров структуры из динамической памяти  delete\_subject(subj3);  delete\_subject(subj4);  return 0;  } |

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



**Приложение. Исходный код программы.**

Файл educational\_subject.h

|  |
| --- |
| #ifndef EDUCATIONAL\_SUBJECT\_H  #define EDUCATIONAL\_SUBJECT\_H  #include <stddef.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <limits.h>  #include <math.h>  #define MAX\_SUBJ\_NAME\_LENGTH 25  #define MAX\_INT\_LENGTH 8  #define MAX\_EXAMFORM\_LENGTH 10  enum attestation\_form  {  Zachet, Exam, Undefined  };  enum error\_code  {  Success, Invalid\_Examination\_Form, Invalid\_Total\_Hours  };  struct discipline  {  char subject\_name[MAX\_SUBJ\_NAME\_LENGTH]; // Название учебного предмета  unsigned short total\_hours; // Общее количество часов  unsigned short lectures\_hours; // Количество часов лекций  unsigned short practice\_hours; // Количество часов практики  enum attestation\_form examination\_form; // Форма аттестации  };  int init\_userdata(struct discipline\* subj);  enum error\_code subject\_DataCheck(const struct discipline\* subj);  enum error\_code subject\_init(struct discipline\* subj, const char\* subject\_name,  unsigned short toral\_hours, unsigned short lectural\_hours,  unsigned short practice\_hours, enum attestation\_form examination\_form);  int struct\_init\_result(enum error\_code result\_code);  void copy\_subject(struct discipline\* dest, const struct discipline\* src);  void subject\_print(struct discipline\* subj);  int field\_changing(struct discipline\* subj);  int update\_Subject\_Name(struct discipline\* subj, const char\* value);  int update\_Total\_Hours(struct discipline\* subj, const unsigned short value);  int update\_Lectures\_Hours(struct discipline\* subj, const unsigned short value);  int update\_Practice\_Hours(struct discipline\* subj, const unsigned short value);  int update\_Examination\_Form(struct discipline\* subj, const enum attestation\_form value);  struct discipline\* create\_subject();  void delete\_subject(struct discipline \*subj);  void input\_string(char\* buffer, int buffer\_size);  int input\_int(const char\* message);  enum attestation\_form input\_attestation\_form(const char\* message);  #endif |

Файл educational\_subject.c

|  |
| --- |
| #include "educational\_subject.h"  int init\_userdata(struct discipline\* subj)  {  char subject\_name[MAX\_SUBJ\_NAME\_LENGTH];  printf("Enter subject name: ");  input\_string(subject\_name, MAX\_SUBJ\_NAME\_LENGTH);  unsigned short total\_hours = input\_int("\nEnter total hours: ");  unsigned short lectures\_hours = input\_int("\nEnter lectures hours: ");  unsigned short practice\_hours = input\_int("\nEnter practice hours: ");  enum attestation\_form examination\_form = input\_attestation\_form("\nEnter examination form (Zachet/Exam): ");    if (struct\_init\_result(subject\_init(subj, subject\_name, total\_hours, lectures\_hours, practice\_hours, examination\_form)) != 0) return 1;  return 0;  }  enum error\_code subject\_DataCheck(const struct discipline\* subj)  {  if (subj->total\_hours < subj->lectures\_hours + subj->practice\_hours)  {  printf("Entered subject hours information is incorrect.\n");  return Invalid\_Total\_Hours;  }  if (subj->examination\_form == Undefined)  {  printf("Entered examination form is incorrect.");  return Invalid\_Examination\_Form;  }  printf("Entered data is correct!\n");  return Success;  }  enum error\_code subject\_init(struct discipline\* subj, const char\* subject\_name, unsigned short total\_hours, unsigned short lectures\_hours, unsigned short practice\_hours, enum attestation\_form examination\_form)  {  strncpy(subj->subject\_name, subject\_name, MAX\_SUBJ\_NAME\_LENGTH);  subj->subject\_name[MAX\_SUBJ\_NAME\_LENGTH - 1] = '\0';  subj->total\_hours = total\_hours;  subj->lectures\_hours = lectures\_hours;  subj->practice\_hours = practice\_hours;  subj->examination\_form = examination\_form;  return subject\_DataCheck(subj);  }  int struct\_init\_result(const enum error\_code result\_code)  {  switch (result\_code)  {  case Success:  printf("Discipline structure initialized successfully.\n");  return 0;  default:  printf("Discipline structure initialization failed.\n");  return 1;  }  }  void copy\_subject(struct discipline \*dest, const struct discipline \*src)  {  strncpy(dest->subject\_name, src->subject\_name, MAX\_SUBJ\_NAME\_LENGTH);  dest->total\_hours = src->total\_hours;  dest->lectures\_hours = src->lectures\_hours;  dest->practice\_hours = src->practice\_hours;  dest->examination\_form = src->examination\_form;  }  void subject\_print(struct discipline\* subj)  {  const char\* form;  switch (subj->examination\_form) {  case Zachet: form = "Zachet"; break;  case Exam: form = "Exam"; break;  default: form = "Undefined"; break;}  printf("Subject - %s . Total hours = %d . Lectures hours = %d . Practice hours = %d . Examination form - %s\n", subj->subject\_name, subj->total\_hours, subj->lectures\_hours, subj->practice\_hours, form);  }  int field\_changing(struct discipline\* subj)  {  int field\_tochange = input\_int("\nWhich field you want to change?\n1 - subject name\n2 - total hours value\n3 - lectures hours value\n4 - practice hours value\n5 - examination form\nEnter any other symbol if you won't change anything\n\nEnter only digit: ");  int init\_checkmark;  switch (field\_tochange)  {  case 1:  char subject\_name\_tochange[MAX\_SUBJ\_NAME\_LENGTH];  input\_string(subject\_name\_tochange, MAX\_SUBJ\_NAME\_LENGTH);  init\_checkmark = struct\_init\_result(update\_Subject\_Name(subj, subject\_name\_tochange));  break;  case 2:  unsigned short total\_hours\_tochange = input\_int("\nEnter new total hours value: ");  init\_checkmark = struct\_init\_result(update\_Total\_Hours(subj, total\_hours\_tochange));  break;  case 3:  unsigned short lectures\_hours\_tochange = input\_int("\nEnter new lectures hours value: ");  init\_checkmark = struct\_init\_result(update\_Lectures\_Hours(subj, lectures\_hours\_tochange));  break;  case 4:  unsigned short practice\_hours\_tochange = input\_int("\nEnter new practice hours value: ");  init\_checkmark = struct\_init\_result(update\_Practice\_Hours(subj, practice\_hours\_tochange));  break;  case 5:  enum attestation\_form examination\_form\_tochange = input\_attestation\_form("\nEnter new attestation form: ");  init\_checkmark = struct\_init\_result(update\_Examination\_Form(subj, examination\_form\_tochange));  break;  default:  printf("\nUndefined value received. Nothing will be changed.\n");  break;  }  if (init\_checkmark == 0) return 0;  return 1;  }  int update\_Total\_Hours(struct discipline\* subj, const unsigned short value)  {  subj->total\_hours = value;  return subject\_DataCheck(subj);  }  int update\_Subject\_Name(struct discipline\* subj, const char\* value)  {  strncpy(subj->subject\_name, value, MAX\_SUBJ\_NAME\_LENGTH);  return subject\_DataCheck(subj);  }  int update\_Lectures\_Hours(struct discipline\* subj, const unsigned short value)  {  subj->lectures\_hours = value;  return subject\_DataCheck(subj);  }  int update\_Practice\_Hours(struct discipline\* subj, const unsigned short value)  {  subj->practice\_hours = value;  return subject\_DataCheck(subj);  }  int update\_Examination\_Form(struct discipline\* subj, const enum attestation\_form value)  {  subj->examination\_form = value;  return subject\_DataCheck(subj);  }  struct discipline\* create\_subject()  {  struct discipline \*subj = (struct discipline \*)malloc(sizeof(struct discipline));    if (!subj) return NULL;  if (init\_userdata(subj) == 0) return (struct discipline \*)subj;  free(subj);  subj = NULL;  return NULL;  }  void delete\_subject(struct discipline \*subj)  {  free(subj);  subj = NULL;  }  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);  }  }  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);  }  enum attestation\_form input\_attestation\_form(const char\* message)  {  char buffer[MAX\_EXAMFORM\_LENGTH];  printf("%s", message);  input\_string(buffer, MAX\_EXAMFORM\_LENGTH);  if (strcmp(buffer, "Zachet") == 0) return Zachet;  if (strcmp(buffer, "Exam") == 0) return Exam;  return Undefined;  } |

Файл main.c

|  |
| --- |
| #include "educational\_subject.h"  int main(int argc, char\* argv[])  {  // Создание экземпляров структур на стеке  struct discipline subj1;  struct discipline subj2;    printf("Gets informaions about educational subject.\n");  printf("Changes it if needs. Outputs result.\n");    // Инициализация экземпляров структуры  if (init\_userdata(&subj1) == 1) return 1;    if (init\_userdata(&subj2) == 1) return 1;    // Вывод сведений о экземплярах структуры  printf("\nFirst subject:\n");  subject\_print(&subj1);  printf("\nSecond subject:\n");  subject\_print(&subj2);  // Инициализация второго экземпляра структуры как копии первого  copy\_subject(&subj2, &subj1);  printf("\nFirst subject:\n");  subject\_print(&subj1);  printf("\nCopy of first subject:\n");  subject\_print(&subj2);  // Вывов функции изменения поля первого экземпляра структуры  if (field\_changing(&subj1) == 1) return 1;  printf("\nResult of changes\n");  subject\_print(&subj1);    // Инициализация двух экземпляров структур в динамической пмяти  printf("Enter data to create first subject.\n");  struct discipline\* subj3 = create\_subject();    if (subj3 == NULL) return 1;    printf("Enter data to create second subject.\n");  struct discipline\* subj4 = create\_subject();  if (subj4 == NULL) return 1;  printf("First: \n");  subject\_print(subj3);  printf("Second: \n");  subject\_print(subj4);    //Инициализируем второй экземпляр структуры как копию первого  copy\_subject(subj4, subj3);    printf("First: \n");  subject\_print(subj3);  printf("Copy of first subject\n");  subject\_print(subj4);  // Вызов функции изменения поля в экземпляре структуры инициализированном в динамической памяти  if (field\_changing(subj3) == 1) return 1;    printf("\nResult of changes\n");  subject\_print(subj3);  // Удаление экземпляров структуры из динамической памяти  delete\_subject(subj3);  delete\_subject(subj4);  return 0;  } |