

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 7**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Линейные динамические списки

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-65-23 |  | Петров А. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

# **Оглавление**

[Оглавление 2](#_Toc166767832)

[1. УСЛОВИЕ ЗАДАЧИ. 2](#_Toc166767833)

[2. ЗАДАНИЕ 1. 5](#_Toc166767834)

[2.1. Разработка абстрактного типа данных задачи, используя шаблон 5](#_Toc166767835)

[2.2. Код алгоритма на языке С++. 7](#_Toc166767836)

[2.3. Тестирование алгоритма. 9](#_Toc166767837)

[3. ЗАДАНИЕ 2. 11](#_Toc166767838)

[3.1. Разработка абстрактного типа данных задачи, используя шаблон 11](#_Toc166767839)

[3.2. Код алгоритма на языке С++ 14](#_Toc166767840)

[3.3. Тестирование алгоритма. 18](#_Toc166767841)

[4. Выводы 22](#_Toc166767842)

[5. Информационные источники 23](#_Toc166767843)

# УСЛОВИЕ ЗАДАЧИ.

**Задание 1**

Разработать и реализовать программу управлению линейным однонаправленным списком в соответствии с условием задачи, определенной вариантом.

Вариант задания включает:

− тип информационной части узла списка;

− условие задачи.

Условие задачи определяет исходные списки (или список) и задачи, которые требуется выполнить над списками (списком)

**Требования к выполнению задания 1**

1. Разработать АТД решения задачи варианта

Включить в раздел данных АТД типы, которые определяют:

− тип информационной части узла;

− тип узла списка;

− указатель на список (или имя типа головного элемента).

В раздел операций включить:

1) операции, общие для всех вариантов:

− вставка узла в начало списка (перед первым узлом);

− создание списка из n узлов, используя функцию вставки узла перед первым

узлом;

− вывод списка в консоль.

2) операции варианта, представленные в условии задачи варианта.

Примечание. Задачи варианта могут быть декомпозированы на вспомогательные

подзадачи, которые также должны быть описаны в АТД.

2. Требования к разработке и реализации задач варианта:

1) каждую задачу АТД реализовать функцией с параметрами,

представляющими входные данные;

2) разработать тесты для каждой задачи варианта.

3. Требования к разработке и реализации программы:

1) программа должна управлять вычислительным процессом посредством

текстового меню, позволяющего выполнить каждую задачу варианта;

2) выполнить тестирование задач варианта на подготовленных тестах.

4. Оформить отчет

**Вариант задачи - № 22**

Дан список, узел которого хранит значение и количество вхождений этого значения (в исходном списке это значение равно1).

1. Вставить символ в список, если этот символ уже есть в списке, то увеличить количество вхождений символа, если такого значения еще нет в списке, то добавить его в конец списка.

2. Удалить из списка узел с символом, который чаще всего встретился в списке.

3. Модифицировать список, переместив первые m узлов в конец списка.

**Задание 2**

Разработать и реализовать программу управлению линейным двунаправленным списком в соответствии с условием задачи, определенной вариантом.

Вариант задания включает:

− тип информационной части узла списка;

− условие задачи.

**Требования к выполнению задания 2**

1. Разработать АТД решения задачи варианта.

1.1. В раздел данных АТД включить типы, которые определяют:

− тип информационной части узла;

− тип узла списка;

− указатель на список (или имя типа головного элемента).

1.2. В раздел операций включить:

1) операции, общие для всех вариантов:

− создание списка из n узлов, используя функцию вставки узла перед первым

узлом;

− вывод списка в двух направлениях (слева направо и справа налево;

− поиск узла с заданным значением ключа (операция должна возвращать

указатель на узел с заданным значением);

2) операции варианта, представленные в условии задачи варианта.

Примечание. Задачи варианта могут быть декомпозированы на вспомогательные

подзадачи, которые также должны быть описаны в АТД.

2. Требования к разработке и реализации задач варианта.

2.1. Каждую задачу АТД реализовать функцией с параметрами,

представляющими входные данные.

2.2. Разработать тесты для каждой задачи постановки задачи варианта.

3. Требования к разработке и реализации программы

3.1. Программа должна управлять вычислительным процессом посредством

текстового меню, позволяющего выполнить каждую задачу варианта;

3.2. Выполнить тестирование задач варианта на подготовленных тестах.

4. Оформить отчет.

**Вариант задачи - № 22**

|  |  |
| --- | --- |
| Тип информационной части узла списка (подчеркнутое поле считать полем ключа) | Условие задачи |
| Инвентарный номер книги, указатель на список областей знаний (к которым относится книга, при создании списка указатель равен NULL), Автор, Название, Год издания.  Примечание. Список областей знаний — это линейный список, информационная часть которого содержит название области знания. | Системный каталог библиотечной автоматизированной системы хранит сведения о книгах фонда в двунаправленном списке. Сведения о принадлежности книги области знания хранит отдельный список, ссылка на который храниться в узле исходного списка соответствующей книги.  Часть списка, начиная с узла с заданным номером (номер больше 1) и до конца списка, перенести в начало списка |

# ЗАДАНИЕ 1.

## 2.1. Разработка абстрактного типа данных задачи, используя шаблон

|  |
| --- |
| АТД MyStruct  {  Данные:  data – Информационная часть узла  count – Количество вхождений символа  next – Указатель на следующий узел |

Операции:

|  |
| --- |
| **Операция 1**  // Функция для вставки узла в начало списка.  //Предусловие: Node\*& head, char value – входные данные.  //Постусловие: Добавляет в список узел и проверяет на наличие такого же значения.  insertNode(Node\*& head, char value). - заголовок |

|  |
| --- |
| **Операция 2**  // Функция для вывода списка в консоль.  //Предусловие: Node\* head – входные данные.  //Постусловие: выводит data каждого узла списка.  printNode(Node\* head). - заголовок |

|  |
| --- |
| **Операция 3**  // Функция для удаления узла с символом, который чаще всего встречается в списке.  //Предусловие: Node\* head – входные данные.  //Постусловие: удаляет узел в списке и назначает новое значение next  deleteMost(Node\*& head). - заголовок |

|  |
| --- |
| **Операция 4:**  // Функция для перемещения первых m узлов в конец списка.  //Предусловие: Node\* head, int m – входные данные.  // Постусловие: перемещает первые m узлов в конец, если же в списке меньше m узлов то возвращает ничего.  moveFirstToEnd(Node\*& head, int m). - заголовок |

## 2.2. Код алгоритма на языке С++.

|  |
| --- |
| #include <iostream>  #include <unordered\_map>  using namespace std;  // Структура для представления узла списка  struct Node {  char data; // Информационная часть узла  int count; // Количество вхождений символа  Node\* next; // Указатель на следующий узел  };  // Функция для вставки узла в начало списка  void insertNode(Node\*& head, char value) {  // Проверяем, существует ли уже узел с данным значением  Node\* current = head;  while (current) {  if (current->data == value) {  // Увеличиваем количество вхождений и выходим  current->count++;  return;  }  current = current->next;  }  // Если символа еще нет в списке, добавляем его в конец  Node\* newNode = new Node{ value, 1, nullptr };  if (!head) {  head = newNode;  }  else {  current = head;  while (current->next) {  current = current->next;  }  current->next = newNode;  }  }  // Функция для вывода списка в консоль  void printNode(Node\* head) {  Node\* current = head;  while (current) {  cout << current->data << "(" << current->count << ") ";  current = current->next;  }  cout << endl;  }  // Функция для удаления узла с символом, который чаще всего встречается в списке  void deleteMost(Node\*& head) {  if (!head) {  return; // Список пуст  }  // Используем хэш-таблицу для подсчета количества вхождений символов  unordered\_map<char, int> countMap;  Node\* current = head;  while (current) {  countMap[current->data]++;  current = current->next;  }  // Находим символ с максимальным количеством вхождений  char mostFrequent = head->data;  int maxCount = countMap[head->data];  current = head->next;  while (current) {  if (countMap[current->data] > maxCount) {  mostFrequent = current->data;  maxCount = countMap[current->data];  }  current = current->next;  }  // Удаляем узел с найденным символом  Node\* prev = nullptr;  current = head;  while (current) {  if (current->data == mostFrequent) {  if (prev) {  prev->next = current->next;  }  else {  head = current->next;  }  delete current;  break;  }  prev = current;  current = current->next;  }  }  // Функция для перемещения первых m узлов в конец списка  void moveFirstToEnd(Node\*& head, int m) {  if (!head || m <= 0) {  return; // Список пуст или некорректное значение m  }  Node\* current = head;  int count = 1;  while (current->next && count < m) {  current = current->next;  count++;  }  if (count < m) {  return; // В списке меньше m узлов  }  // Находим последний узел в текущем списке  Node\* lastNode = current;  while (lastNode->next) {  lastNode = lastNode->next;  }  // Перемещаем первые m узлов в конец списка  lastNode->next = head;  head = current->next;  current->next = nullptr;  }  int main() {  setlocale(LC\_ALL, "Russian");  Node\* head = nullptr;  // Вставляем символы в список  insertNode(head, 'a');  insertNode(head, 'b');  insertNode(head, 'c');  insertNode(head, 'b');  insertNode(head, 'd');  insertNode(head, 'c');  insertNode(head, 'b');  insertNode(head, 'a');  insertNode(head, 'c');  insertNode(head, 'd');  insertNode(head, 'e');  insertNode(head, 'f');  // Выводим список в консоль  cout << "Оригинальный список: ";  printNode(head);  // Удаляем узел с символом, который чаще всего встречается в списке  deleteMost(head);  cout << "Список после удаления наиболее часто встречающегося узла: ";  printNode(head);  // Перемещаем первые m узлов в конец списка  int m = 3;  moveFirstToEnd(head, m);  cout << "Список после первого перемещения " << m << " узлы до конца: ";  printNode(head);  return 0;  } |

## 2.3. Тестирование алгоритма.

Таблица 1 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| Тестирование алгоритма для задания 1 | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | insertNode(head, 'a');  insertNode(head, 'b');  insertNode(head, 'c');  insertNode(head, 'b');  insertNode(head, 'd');  insertNode(head, 'c');  insertNode(head, 'b');  insertNode(head, 'a');  insertNode(head, 'c');  insertNode(head, 'd');  insertNode(head, 'e');  insertNode(head, 'f'); printNode(head deleteMost(head) printNode(head) moveFirstToEnd(head, m) printNode(head) | Оригинальный список:  a(2) b(3) c(3) d(2) e(1) f(1)  Список после удаления наиболее часто встречающегося узла:  b(3) c(3) d(2) e(1) f(1)  Список после первого перемещения 3 узлы до конца:  e(1) f(1) b(3) c(3) d(2) |



Рисунок 1 - Скриншот тестирования

# ЗАДАНИЕ 2.

## 3.1. Разработка абстрактного типа данных задачи, используя шаблон

|  |
| --- |
| АТД KnowledgeAreaNode  {  Данные:  Name - Название области знания  next – Указатель на следующий узел |
| АТД BookNode  {  Данные:  inventoryNumber - Инвентарный номер книги (ключ)  author - Автор книги  title - Название книги  year - Год издания  knowledgeAreas - Указатель на список областей знаний  prev - Указатель на предыдущий узел двунаправленного списка  next - Указатель на следующий узел двунаправленного списка |

Операции:

|  |
| --- |
| **Операция 1**  // Функция для создания нового узла списка областей знаний.  //Предусловие: name – входные данные.  //Постусловие: создает новый объект типа KnowledgeAreaNode – область знаний  createKnowledgeAreaNode(const string& name) - заголовок |

|  |
| --- |
| **Операция 2**  // Функция для создания нового узла двунаправленного списка книг.  //Предусловие: inventoryNumber, author, title, year, knowledgeAreas – входные данные.  //Постусловие: создает новый объект BookNode – книгу.  createBookNode(int inventoryNumber, const string& author, const string& title, int year, KnowledgeAreaNode\* knowledgeAreas) - заголовок |

|  |
| --- |
| **Операция 3**  // Функция для добавления новой области знания в список областей знаний книги.  //Предусловие: book, areaName – входные данные.  //Постусловие: Создает новую область знаний в списке  addKnowledgeArea(BookNode\* book, const string& areaName) - заголовок |

|  |
| --- |
| **Операция 4:**  // Функция для создания двунаправленного списка книг из n узлов  //Предусловие: n – входные данные.  // Постусловие: создает список из n узлов  createBookList(int n). - заголовок |

|  |
| --- |
| **Операция 5:**  // Функция для вывода списка книг в двух направлениях  //Предусловие: head – входные данные.  // Постусловие: выводит в консоль список книг слева направо и справа налево  printBookList(const BookNode\* head) - заголовок |

|  |
| --- |
| **Операция 6:**  // Функция для поиска узла с заданным инвентарным номером книги  //Предусловие: head, n – входные данные.  // Постусловие: выводит в консоль название книги  findBookByInventoryNumber(const BookNode\* head, int inventoryNumber) - заголовок |

|  |
| --- |
| **Операция 7:**  // Функция для создания двунаправленного списка книг из n узлов  //Предусловие: head, n – входные данные.  // Постусловие: выводит в консоль всю информацию о книге  printBookInfoByInventoryNumber(const BookNode\* head, int inventoryNumber). - заголовок |

|  |
| --- |
| **Операция 8:**  // Функция для вывода областей знаний и книг, которые к ним принадлежат  //Предусловие: head – входные данные.  // Постусловие: выводит в консоль все области знаний и книги  printKnowledgeAreasAndBooks(const BookNode\* head). - заголовок |

|  |
| --- |
| **Операция 9:**  // Функция для перемещения части списка, начиная с заданного узла, в начало списка  //Предусловие: head, startInventoryNumber– входные данные.  // Постусловие: перемещает книги в начало списка начиная с позиции startInventoryNumber  printBookInfoByInventoryNumber(const BookNode\* head, int inventoryNumber). - заголовок |

## 3.2. Код алгоритма на языке С++

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  // Структура для представления узла списка областей знаний  struct KnowledgeAreaNode {  string name; // Название области знания  KnowledgeAreaNode\* next; // Указатель на следующий узел списка областей знаний  };  // Структура для представления узла двунаправленного списка книг  struct BookNode {  int inventoryNumber; // Инвентарный номер книги (ключ)  string author; // Автор книги  string title; // Название книги  int year; // Год издания  KnowledgeAreaNode\* knowledgeAreas; // Указатель на список областей знаний  BookNode\* prev; // Указатель на предыдущий узел двунаправленного списка  BookNode\* next; // Указатель на следующий узел двунаправленного списка  };  // меню  void displayMenu(BookNode\*& bookList);  // Функция для создания нового узла списка областей знаний  KnowledgeAreaNode\* createKnowledgeAreaNode(const string& name) {  return new KnowledgeAreaNode{ name, nullptr };  }  // Функция для создания нового узла двунаправленного списка книг  BookNode\* createBookNode(int inventoryNumber, const string& author, const string& title, int year, KnowledgeAreaNode\* knowledgeAreas) {  return new BookNode{ inventoryNumber, author, title, year, knowledgeAreas, nullptr, nullptr };  }  // Функция для добавления новой области знания в список областей знаний книги  void addKnowledgeArea(BookNode\* book, const string& areaName) {  KnowledgeAreaNode\* newNode = createKnowledgeAreaNode(areaName);  if (!book->knowledgeAreas) {  book->knowledgeAreas = newNode;  }  else {  KnowledgeAreaNode\* current = book->knowledgeAreas;  while (current->next) {  current = current->next;  }  current->next = newNode;  }  }  // Функция для создания двунаправленного списка книг из n узлов  BookNode\* createBookList(int n) {  BookNode\* head = nullptr;  for (int i = 0; i < n; ++i) {  int inventoryNumber;  string author, title;  int year;  // Пример для демонстрации:  inventoryNumber = i + 1;  author = "Author" + to\_string(i + 1);  title = "Title" + to\_string(i + 1);  year = 2000 + i;  // Создаем узел книги и добавляем его в список  BookNode\* newNode = createBookNode(inventoryNumber, author, title, year, nullptr);  if (!head) {  head = newNode;  }  else {  BookNode\* current = head;  while (current->next) {  current = current->next;  }  current->next = newNode;  newNode->prev = current;  }  }  return head;  }  // Функция для вывода списка книг в двух направлениях  void printBookList(const BookNode\* head) {  const BookNode\* current = head;  cout << "Книги слева направо: ";  while (current) {  cout << current->inventoryNumber << " ";  current = current->next;  }  cout << endl;  // Выводим список в обратном порядке  cout << "Книги справа налево: ";  current = head;  while (current->next) {  current = current->next;  }  while (current) {  cout << current->inventoryNumber << " ";  current = current->prev;  }  cout << endl;  }  // Функция для поиска узла с заданным инвентарным номером книги  BookNode\* findBookByInventoryNumber(const BookNode\* head, int inventoryNumber) {  const BookNode\* current = head;  while (current) {  if (current->inventoryNumber == inventoryNumber) {  return const\_cast<BookNode\*>(current);  }  current = current->next;  }  return nullptr;  }  // Функция для вывода информации о книге по её инвентарному номеру  void printBookInfoByInventoryNumber(const BookNode\* head, int inventoryNumber) {  BookNode\* book = findBookByInventoryNumber(head, inventoryNumber);  if (book) {  cout << "=== Book Information ===" << endl;  cout << "Inventory Number: " << book->inventoryNumber << endl;  cout << "Author: " << book->author << endl;  cout << "Title: " << book->title << endl;  cout << "Year: " << book->year << endl;  }  else {  cout << "Book not found" << endl;  }  }  // Функция для вывода областей знаний и книг, которые к ним принадлежат  void printKnowledgeAreasAndBooks(const BookNode\* head) {  // Проходим по списку книг и выводим информацию о каждой книге и её областях знаний  const BookNode\* currentBook = head;  while (currentBook) {  cout << "=== Book Information ===" << endl;  cout << "Title: " << currentBook->title << endl;  cout << "Author: " << currentBook->author << endl;  cout << "Year: " << currentBook->year << endl;  cout << "Knowledge Areas: ";  KnowledgeAreaNode\* currentKnowledge = currentBook->knowledgeAreas;  while (currentKnowledge) {  cout << currentKnowledge->name << ", ";  currentKnowledge = currentKnowledge->next;  }  cout << endl;  currentBook = currentBook->next;  }  }  // Функция для перемещения части списка, начиная с заданного узла, в начало списка  void movePartOfListToStart(BookNode\*& head, int startInventoryNumber) {  // Находим узел, с которого нужно начать перемещение  BookNode\* startNode = findBookByInventoryNumber(head, startInventoryNumber);  if (!startNode || !startNode->next) {  return; // Узел не найден или нет узлов после него  }  // Находим последний узел в текущем списке  BookNode\* lastNode = startNode;  while (lastNode->next) {  lastNode = lastNode->next;  }  // Перемещаем часть списка в начало  lastNode->next = head;  head->prev = lastNode;  head = startNode->next;  head->prev = nullptr;  startNode->next = nullptr;  }  int main() {  setlocale(LC\_ALL, "Russian");  // Создаем список книг  int n = 5; // Количество книг  BookNode\* bookList = createBookList(n);  // Добавляем 1 область знаний для книг для теста  BookNode\* currentBook = bookList;  while (currentBook) {  addKnowledgeArea(currentBook, "Область знаний 1");  currentBook = currentBook->next;  }  while (true) {  displayMenu(bookList);  }  return 0;  }  // Функция для вывода меню и обработки выбора пользователя  void displayMenu(BookNode\*& bookList) {  char choice;  int inventoryNumber;  cout << "===== Меню =====" << endl;  cout << "1. Вывести список книг" << endl;  cout << "2. Найти книгу по инвентарному номеру" << endl;  cout << "3. Переместите часть списка в начало" << endl;  cout << "4. Вывести информацию о книге по ее номеру" << endl;  cout << "5. Вывести информацию о областях знаний" << endl;  cout << "6. Выход" << endl;  cout << "Введите свой выбор: ";  cin >> choice;  switch (choice) {  case '1':  cout << "=== Список книг ===" << endl;  printBookList(bookList);  break;  case '2':  cout << "Введите инвентарный номер для поиска: ";  cin >> inventoryNumber;  {  BookNode\* foundBook = findBookByInventoryNumber(bookList, inventoryNumber);  if (foundBook) {  cout << "Найденная книга: " << foundBook->title << endl;  }  else {  cout << "Книга не найдена" << endl;  }  }  break;  case '3':  cout << "Введите инвентарный номер, с которого вы начнете перемещение: ";  cin >> inventoryNumber;  movePartOfListToStart(bookList, inventoryNumber);  cout << "Часть списка перемещена для успешного запуска" << endl;  break;  case '4':  cout << "Введите инвентарный номер книги: ";  cin >> inventoryNumber;  printBookInfoByInventoryNumber(bookList, inventoryNumber);  break;  case '5':  cout << "=== Области знаний и книги ===" << endl;  printKnowledgeAreasAndBooks(bookList);  break;  case '6':  cout << "выход из программы" << endl;  exit(0);  default:  cout << "Неверный выбор" << endl;  break;  }  } |

## 3.3. Тестирование алгоритма.

Таблица 2 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| Тестирование алгоритма для задания 2 | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | printBookList(bookList) | === Список книг ===  Книги слева направо:  1 2 3 4 5  Книги справа налево:  5 4 3 2 1 |
| 2 | findBookByInventoryNumber(bookList, 2) | Найденная книга: Title2 |
| 3 | movePartOfListToStart(bookList, 3) | Книги слева направо:  4 5 1 2 3  Книги справа налево:  3 2 1 5 4 |
| 4 | printBookInfoByInventoryNumber(bookList, 1) | === Book Information ===  Inventory Number: 1  Author: Author1  Title: Title1  Year: 2000 |
| 5 | printKnowledgeAreasAndBooks(bookList) | === Области знаний и книги ===  === Book Information ===  Title: Title4  Author: Author4  Year: 2003  Knowledge Areas: Область знаний 1,  === Book Information ===  Title: Title5  Author: Author5  Year: 2004  Knowledge Areas: Область знаний 1,  === Book Information ===  Title: Title1  Author: Author1  Year: 2000  Knowledge Areas: Область знаний 1,  === Book Information ===  Title: Title2  Author: Author2  Year: 2001  Knowledge Areas: Область знаний 1,  === Book Information ===  Title: Title3  Author: Author3  Year: 2002  Knowledge Areas: Область знаний 1, |

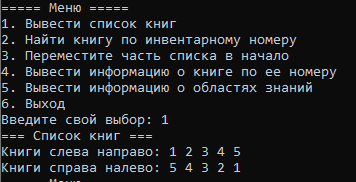


Рисунок 2 - Скриншот тестирования № 1

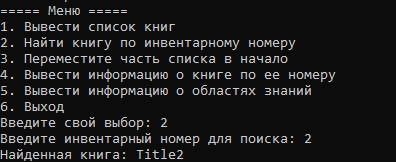


Рисунок 3 - Скриншот тестирования № 2

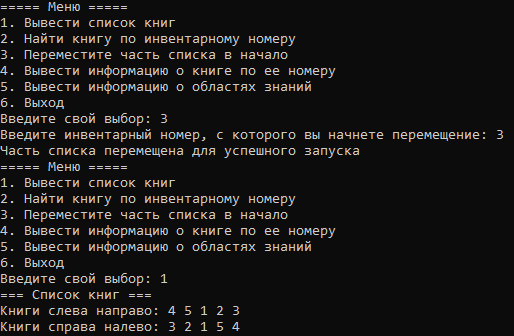


Рисунок 4 - Скриншот тестирования № 3

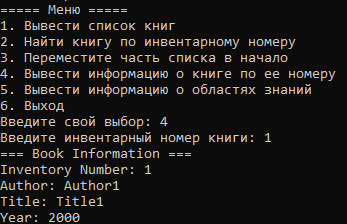


Рисунок 5 - Скриншот тестирования № 4

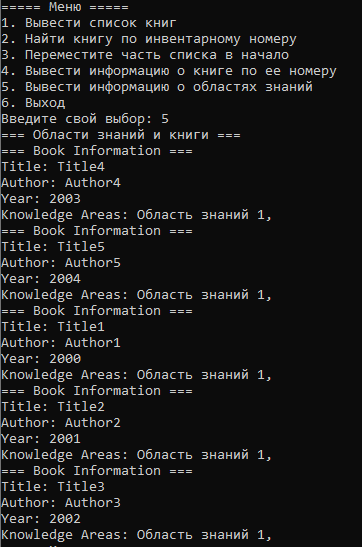


Рисунок 6 - Скриншот тестирования № 5

# 4. Выводы

Во время выполнения задания 1 на пятой практической работе, я изучил библиотеку string.h, работу с динамическими массивами \*\*char и преобразование строк в нулевые строки. Однако, когда я приступил к выполнению задания 2, я освоил современные методы работы со строками, добавленные в стандарт C++, и пришёл к выводу, что использование этих современных методов делает код гораздо более читаемым и компактным.

# 5. Информационные источники

1. Приложение к практическим работам – СДО (online-edu.mirea.ru)

2. Структуры и алгоритмы обработки данных – методические указания / Скворцова Л.А., Филатов А.С., Гусев К.В., Трушин С.М. - Москва, МИРЭА - Российский технологический университет, 2023.