**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**«Казанский национальный исследовательский технический университет**

**им. А.Н. Туполева-КАИ»**

Институт Компьютерных технологий и защиты информации

Кафедра Прикладной математики и информатики имени Ю.В. Кожевникова

Пояснительная записка

к курсовой работе

по дисциплине «Структуры и алгоритмы обработки данных»

на тему «Реализация комбинированных структур данных»

вариант №54.10

**Выполнил**

Студент группы 4218

Данилаева С.Д.

**Проверил**

Доцент кафедры ПМИ

Сотников Сергей Викторович

Казань 2023

Оглавление

[Глава 1. Постановка задачи 3](#_Toc136831785)

[Глава 2. Теоретическое описание разработанной структуры 4](#_Toc136831786)

[2.1. Логическая составляющая структуры 4](#_Toc136831787)

[2.2. Набор основных операций 4](#_Toc136831788)

[2.3. Алгоритмизация основных операций 5](#_Toc136831789)

[2.4. Реализация структуры 6](#_Toc136831790)

[Глава 3. Описание программного комплекса 9](#_Toc136831791)

[3.1. Описание структуры проекта 9](#_Toc136831792)

[3.2. Руководство программиста 9](#_Toc136831793)

[3.2.1. Описание разработанных классов 9](#_Toc136831794)

[3.2.2. Описание структуры внешнего файла 11](#_Toc136831795)

[3.3. Руководство пользователя 12](#_Toc136831796)

[3.3.1. Описание демонстрационного модуля 12](#_Toc136831797)

[Заключение 17](#_Toc136831798)

[Литература 18](#_Toc136831799)

[Приложения 19](#_Toc136831800)

[Приложение 1. Листинг Box\_func.cpp 19](#_Toc136831801)

[Приложение 2. Листинг Section\_func.cpp 19](#_Toc136831802)

[Приложение 3. Листинг Stock\_func.cpp 21](#_Toc136831803)

[Приложение 4. Листинг Menu\_func.cpp 24](#_Toc136831804)

[Приложение 5. Листинг Kursach.cpp 25](#_Toc136831805)

[Приложение 6. Листинг Menu.h 26](#_Toc136831805)

[Приложение 7. Листинг Skald.h 26](#_Toc136831805)

# Глава 1. Постановка задачи

Цель работы: Реализовать библиотеку классов, реализующих контейнеры, инкапсулирующие комбинированную структуру данных типа «статический неупорядоченный список динамических очередей» с информационным наполнением «склад (наименование) – композиция секций(номер), секция – композиция ячеек (номер, признак занятости)».

Требования к разрабатываемой программе:

1) Реализация всех необходимых операций (добавление и удаление в основной и присоединённой структурах, поиск в списке).

2) Возможность сохранения всей структуры во внешнем файле (текстовом или XML) с обратной загрузкой.

3) Реализация структуры для хранения и обработки данных конкретной информационной задачи.

4) Именование типов, структур и их полей, классов и их свойств и методов в соответствии с конкретной информационной задачей.

5) Наличие демо-модуля с удобным оконным пользовательским интерфейсом.

Требования к программной документации:

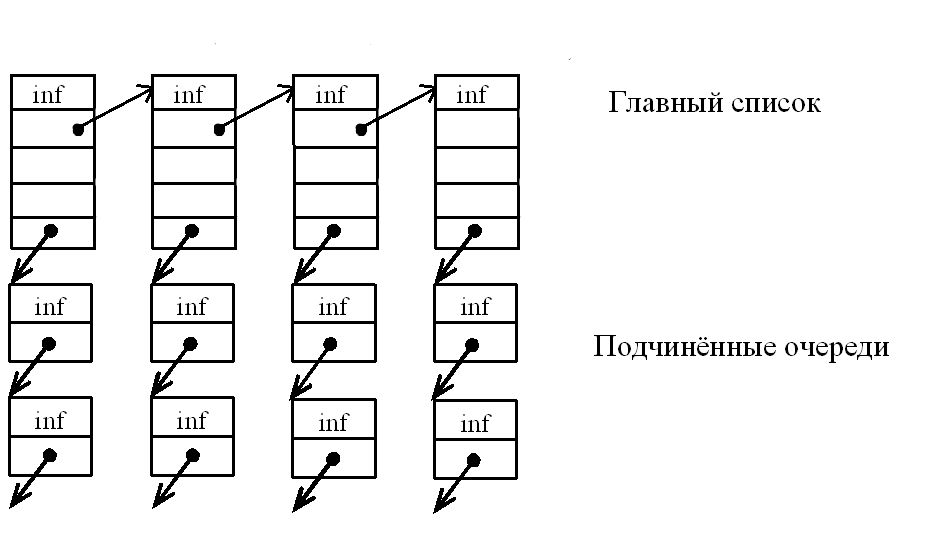
1) Описание всех функций каждого модуля (для чего они предназначены, их входные и выходные параметры, тип и назначение).

2) Описание возможностей, предоставляемых данным программным комплексом (интерфейсная часть).

# Глава 2. Теоретическое описание разработанной структуры

## 2.1. Логическая составляющая структуры

Схематически комбинированная структура «статический список очередей» выглядит следующим образом:



## 2.2. Набор основных операций

Все операции комбинированной структуры делятся на 3 типа: операции над главной структурой, операции над подчинённой структурой и обход всей структуры.

Операции над главным списком :

1. Инициализация
2. Проверка возможности добавления
3. Проверка возможности удаления
4. Обход списка
5. Поиск элемента
6. Добавление элемента
7. Удаление элемента

Операции над подчинённой структурой:

1. Инициализация
2. Проверка возможности удаления:
3. Обход списка
4. Поиск элемента
5. Добавление элемента

## 2.3. Алгоритмизация основных операций

Операции над главным списком:

1. Инициализация

2. Проверка возможности добавления

* Проверка заполненности списка и добавление элемента

3. Проверка возможности удаления

* Проверка не пустоты списка и удаление элемента

4. Обход главного списка

* Задание начального значения
* В цикле, пока в списке остались необработанные элементы
* Обработка текущего элемента
* Переход к следующему элементу

5. Поиск элемента в главном списке

* Проверка возможности поиска
* Запрос информационной части искомого элемента
* Задание начального значения
* В цикле, пока в списке остались необработанные элементы и нужный элемент не найден
* Если информационная часть текущего элемента удовлетворяет заданному критерию, то определить текущий элемент как искомый (найденный) и выйти из цикла, вернув указатель на найденный элемент
* Иначе – переход к следующему элементу
* При выходе из цикла сообщить о том, что поиск был неудачен (передать пустой указатель)

6. Добавление элемента очереди

* Проверка возможности добавления
* Определение положения добавляемого элемента (задание последующего и его поиск)
* Запрос информационной части нового элемента
* Определение (инициализация полей) нового элемента

7. Удаление элемента очереди

* Проверка возможности удаления
* Задание удаляемого элемента (его информационной части)
* Поиск удаляемого элемента
* Обработка удаляемого элемента

Операции над подчинённым списком:

Отличие алгоритмизации заключается в том, что в начале каждой операции 1-м шагом осуществляется поиск подчинённого списка в главном, а затем уже все остальные шаги. Проверка на возможность добавления не нужна.

Остальные операции работают так же.

Обход всей структуры:

Обход всей структуры включает в себя:

* Обход главного списка
* При обработке элементов главного списка осуществлять обход соответствующих подчинённых структур

## 2.4. Реализация структуры

Согласно заданию был выбран статический метод реализации для главной, и динамический для подчинённой структур.

В соответствии с заданным типом реализации для хранения данных главной структуры будет использоваться массив определенного размера.

Для реализации структуры потребуем следующее:

*class Stock*

*private:*

*std::string name;*

*Section sect[MAX]{};*

*int count*

*class Section*

*private:*

*int section\_id;*

*Box\* head;*

*Box\* tail;*

*int count;*

Операции над главным списком:

1. Инициализация главного списка
2. Проверка возможности добавления

stock->Is\_Full();

3. Проверка возможности удаления

stock-> Is\_Empty();

4. Обход главного списка

stock->Stock\_status();

5. Поиск заданного элемента

stock->Find\_section(int section\_id);

6. Добавление элемента

stock->Add\_section(int section\_id);

7. Удаление элемента

stock->Delete\_section(int section\_id);

Операции над подчинённой структурой:

1. Инициализация

2. Проверка возможности удаления

section->Is\_Empty();

3. Добавление элемента очереди

section->Add\_box(int box\_id, bool box\_occupancy);

4. Удаление элемента из очереди

section->Delete\_box();

5. Обход очереди

section->Section\_status();

1. Поиск элемента очереди

section->Find\_box(int bx\_id);

Обход всей структуры:

if (!Is\_Empty()) {

std::cout << "Наименование склада: " << name << "\n";

for (int i = 0; i < count; i++) {

std::cout << " id секции = " << sect[i].Get\_section\_id() << "\n";

sect[i].Section\_status();

}

}

# Глава 3. Описание программного комплекса

## 3.1. Описание структуры проекта

Для создания проекта была использована среда разработки Microsoft Visual Studio 2022. Проект имеет следующую иерархию:

Kursach.cpp

Kursach.h

Stock\_func.cpp

Section\_func.cpp

Box\_func.cpp

Sklad.h

Menu\_func.cpp

Menu.h

output.txt

input.txt

Kursach – основная папка проекта.

Kursach.cpp – файл, содержащий исходный код демонстрационного модуля проекта и точку входа;

\*.h – файлы-заголовков, содержащие описание созданных классов без реализации их методов;

\*.cpp – файлы, содержащий реализацию методов классов, описанных в файле-заголовке \*.h;

\*.txt – файлы, создаваемые пользователем для хранения данных.

## 3.2. Руководство программиста

### 3.2.1. Описание разработанных классов

Для реализации комбинированной структуры было разработано 3 класса: класс Stock, реализующий поведение главной структуры, класс Section, реализующий поведение подчинённой структуры, и класс Box, реализующий поведение элемента нижнего уровня. Дополнительно был реализован класс Menu для удобства создания функций меню.

* + - 1. **Описание класса Stock**

Поля:

std::string name – название склада

Section sect[MAX]{} – массив секций (динамических очередей)

int count – количество секций

|  |  |
| --- | --- |
| Метод класса | Описание метода |
| Stock(); | Создаёт заголовок списка, устанавливает значение поля  name и count стандартным значением. |
| void Set\_name(std::string n); | Устанавливает наименование склада. |
| bool Is\_Empty(); | Проверяет, пуст ли список с секциями. |
| bool Is\_Full(); | Проверяет, заполнен ли список с секциями. |
| void Add\_section(int section\_id); | Добавляет секцию. |
| int Find\_section(int section\_id); | Осуществляет поиск секции с заданным номером. |
| void Stock\_status\_F(const std::string filename); | Производит заполнение информации о складе из заданного файла. |
| void Stock\_printf(const std::string filename); | Записывает текущее состояние склада в заданный файл. |
| void Stock\_status(); | Выводит текущее состояние склада в консоль. |
| void Delete\_section(int section\_id); | Удаляет заданную секцию. |
| void Delete\_stock(); | Удаляет полностью всю информацию о складе. |
| Section\* Ret\_section(int section\_id); | Возвращает указатель секцию с заданным номером. |

* + - 1. **Описание класса Section**

Поля:

int section\_id – номер секции.

Box\* head – указатель на начало очереди.

Box\* tail– указатель на конец очереди.

int count – количество ячеек

|  |  |
| --- | --- |
| Метод класса | Описание метода |
| Section(); | Устанавливает значение поля  section\_id и count стандартным значением. |
| Section(int id); | Устанавливает значение поля  section\_id определенным и count стандартным значением. |
| void Set\_section(int id); | Устанавливает определенное значение поля section\_id. |
| int Get\_section\_id(); | Возвращает значение поля section\_id. |
| int Get\_box\_count(); | Возвращает значение поля сount. |
| Box\* Get\_section\_head(); | Возвращает значение поля head. |
| bool Is\_Empty(); | Проверяет пуста ли секция. |
| void Add\_box(int box\_id, bool box\_occupancy); | Добавляет ячейку с заданными параметрами. |
| void Section\_status(); | Выводит текущее состояние секции в консоль. |
| void Delete\_box(); | Удаляет ячейку. |
| int Find\_box(int bx\_id); | Осуществляет поиск заданной ячейки. |
| Box\* Ret\_box(int bx\_id); | Возвращает указатель на ячейку с указанным номером. |

**3.2.1.3. Описание класса Box**

Поля:

int box\_id – номер ячейки

bool occupancy – статус заполненности

Box\* next – указатель на следующий элемент

|  |  |
| --- | --- |
| Метод класса | Описание метода |
| Box(); | Создаёт ячейку со стандартными значениями полей. |
| Box(int id, bool oc); | Cоздаёт ячейку с заданными параметрами. |
| void Set\_id(int id); | Устанавливает значение поля box\_id. |
| int Get\_id(); | Возвращает значение поля box\_id. |
| void Set\_occupancy(bool occupancy); | Устанавливает значение поля occupancy. |
| void Set\_next(Box\* nex); | Устанавливает значение поля next. |
| bool Get\_occupancy(); | Возвращает значение поля occupancy. |
| Box\* Get\_next(); | Возвращает значение поля next. |

### 3.2.2. Описание структуры внешнего файла

**3.2.2.1. Информация о записи и загрузке данных с использованием внешнего файла**

Для хранения данных о комбинированной структуре во внешних файлах и последующей загрузки этих данных в основную программу будет

использоваться обычный текстовый файл.

Сам файл имеет следующую структуру:

* + - * 1. На первой строке в файле находится название склада
        2. Вторая и третья строки имеют вид:

Section count: – количество секций на складе

Section numbers: – перечисленные через пробел номера всех секций

* + - * 1. Четвертая и последующие строки имеют вид:

Box count for (номер секции): – общее количество ячеек в секции

Box numbers and status for (номер секции): – номера и статусы ячеек перечисленные через пробел в виде номер\_ячейки-статус\_заполенности

**3.2.2.2. Общий вид структуры внешнего файла**

Таким образом, с учётом всех вышеописанных правил, структура файла имеет следующий вид:

Stock name: (наименование склада)

Section count: (количество секций)

Section numbers: (существующие секции)

Box count for (номер секции): (количество ячеек в секции)

Box numbers and status for (номер секции): (номер ячейки)-(статус заполненности)

## 3.3. Руководство пользователя

### 3.3.1. Описание демонстрационного модуля

**3.1.1.1. Описание работы с демонстрационным модулем**

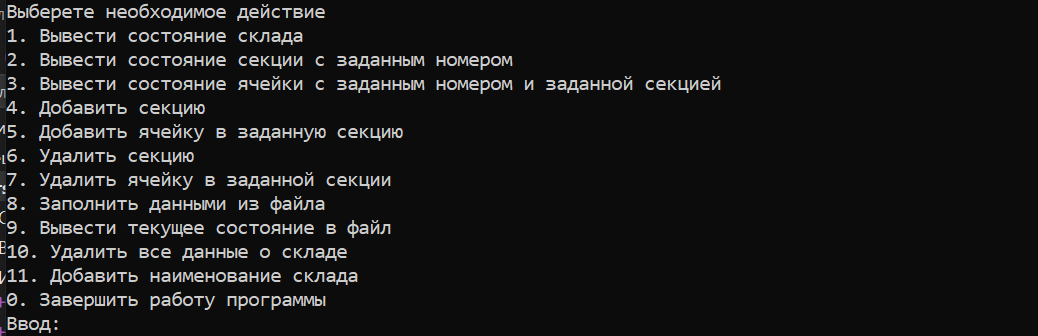
После запуска приложения откроется окно консоли, представляющее собой демонстрационный модуль для работы с разработанными классами.

Взаимодействие с пользователем посредством консоли осуществляется следующим образом: пользователю предлагается меню – пронумерованный список команд – из которого требуется выбрать одну команду и указать её номер, а далее, следуя требованиям, указать дополнительную информацию, требуемую для выполнения той или иной программы.

При вводе пользователем тех или иных данных программа проверяет их соответствие требуемому типу данных и адекватность. Если данные корректные, программа продолжает выполнение – в противном случае пользователю сообщается о некорректности введённых данных, после чего предоставляется ещё одна попытка ввода. Программа даёт бесконечное количество попыток ввода до тех пор, пока ввод не окажется корректен.

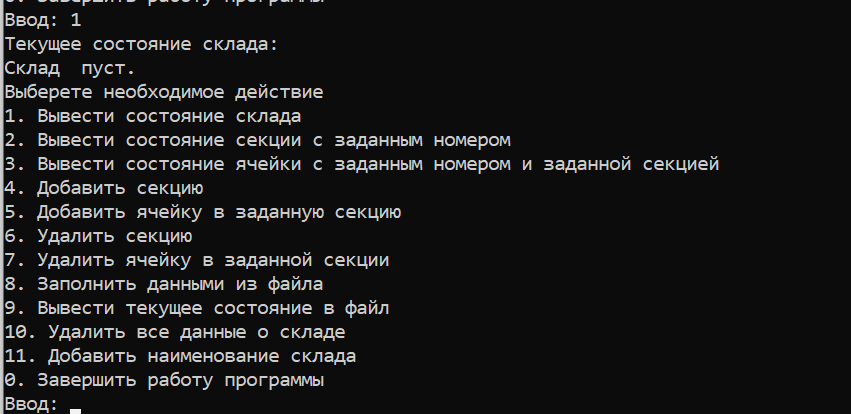
**3.1.1.2. Меню создания Программы**

После запуска программы перед пользователем откроется консольное меню:



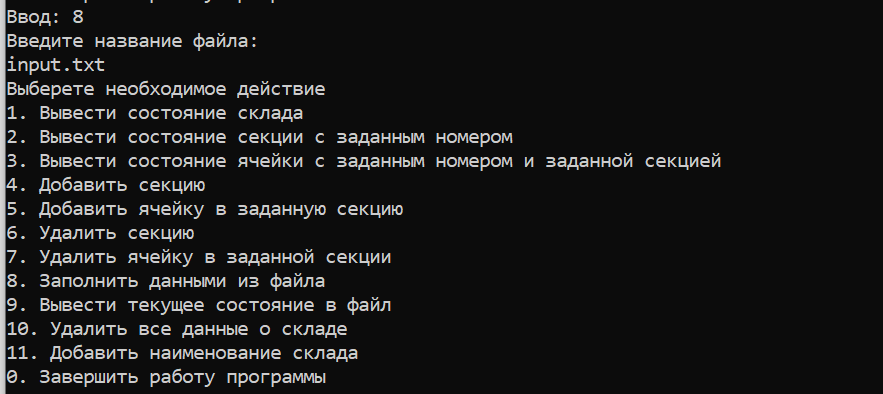
Необходимо выбрать нужную команду и ввести ее с клавиатуры.

Выбрав пункт вывести состояние склада, мы получим сообщение о том, что склад пуст, так как мы его еще не заполнили:

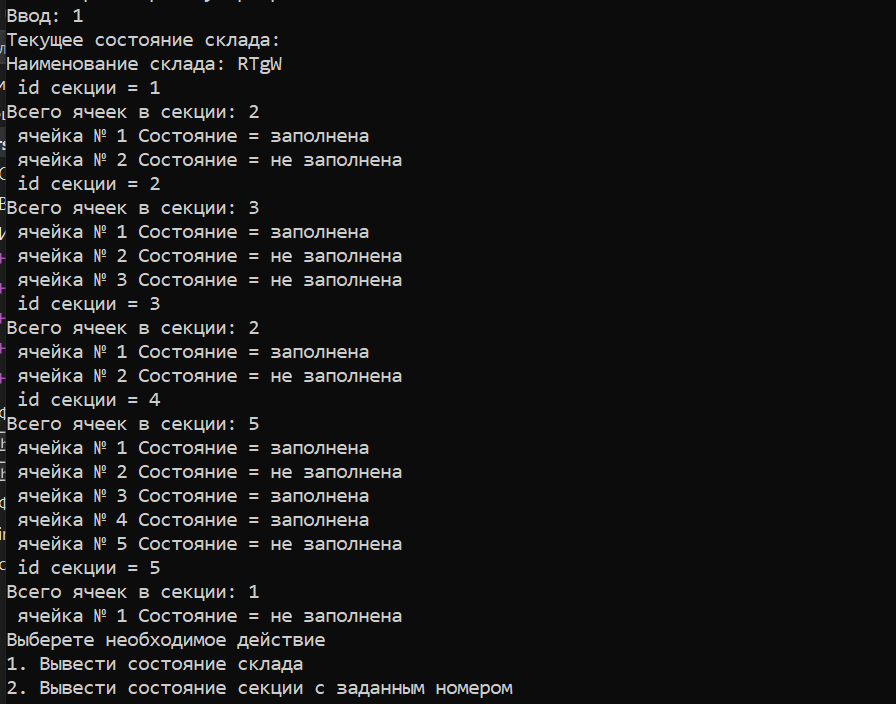
****

При попытке, к примеру, удалить секцию или найти секцию на складе, мы получим сообщение о том, что склад пуст, поэтому склад необходимо заполнить.

Мы можем выгрузить заранее заготовленную структуру склада из файла «input.txt», с помощью следующих команд:



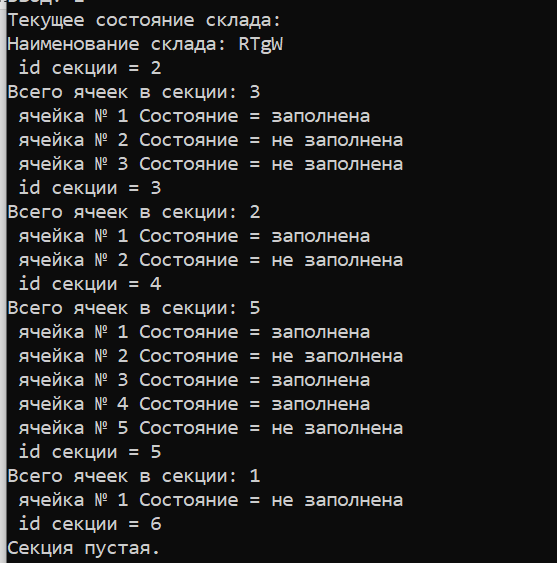
Выведем состояние склада:



Теперь мы можем работать со структурой склада. Например добавлять и удалять ячейки и секции. Для добавления секции выбираем 4 пункт меню, после вводим номер секции.

Для удаления выбираем 6 пункт меню, а после вводим номер секции.

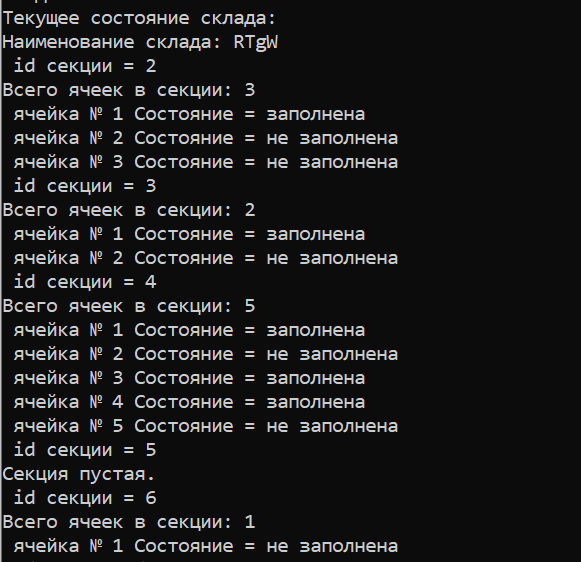
При удалении секции 1 и добавлении секции 6 получим:



Для добавления ячейки сначала необходимо выбрать 5 пункт меню, ввести номер секции, затем номер ячейки и статус ее заполненности.

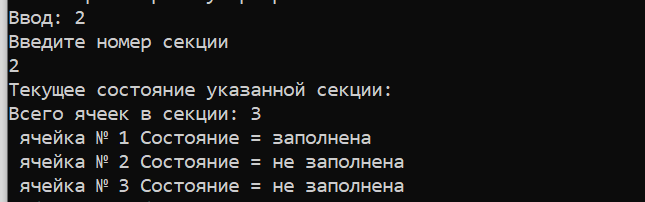
Для удаления ячейки необходимо выбрать 7 пункт меню, после ввести номер секции.

При добавлении в секцию 6 пустой ячейки 1 и удалении ячейки из 5 секции получим:

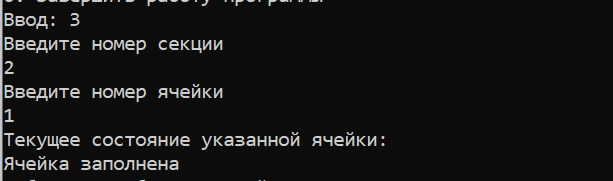


Также можно произвести поиск необходимой ячейки или секции с выводом ее состояния.

Поиск секции:

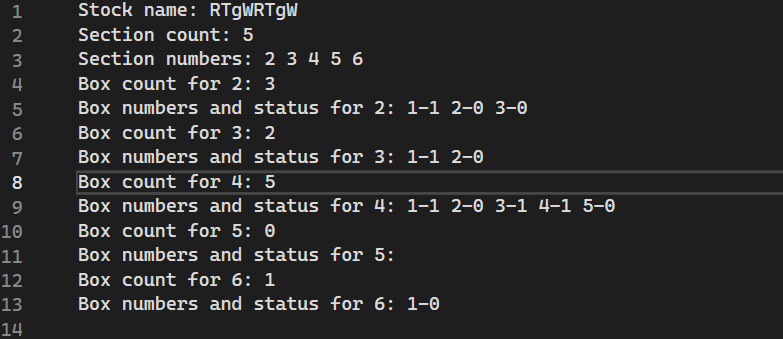


Поиск ячейки:



Текущее состояние склада можно вывести в файл командой номер 10.

На данный момент в файл запишется следующая информация:



Изменить наименование склада можно командой 11.

Команда 10 произведет полное удаление информации о складе.

# Заключение

В результате выполнения курсовой работы была реализована заданная комбинированная структура «Статический неупорядоченный список динамических очередей» с информационным наполнением вида склад (наименование) – композиция секций(номер), секция – композиция ячеек (номер, признак занятости) в виде библиотеки разработанных классов. Был создан пользовательский интерфейс, демонстрирующий работу с заданной структурой. Кроме этого, были разработаны руководство разработчика и руководство пользователя по работе с данным приложением.

# Литература

1. Козин А. Н. Структуры и алгоритмы обработки данных. Учебно-методическое пособие. Издательство: Татарский институт содействия бизнесу, 2003. – 193с.

# Приложения

## Приложение 1. Листинг Box\_func.cpp

#include"Sklad.h"

Box::Box() {

box\_id = -1;

occupancy = 0;

next = nullptr;

}

Box::Box(int id, bool oc) {

box\_id = id;

occupancy = oc;

next = nullptr;

}

void Box::Set\_id(int id) {

box\_id = id;

}

int Box::Get\_id() {

return box\_id;

}

void Box::Set\_occupancy(bool new\_occupancy) {

occupancy = new\_occupancy;

}

bool Box::Get\_occupancy() {

return occupancy;

}

void Box::Set\_next(Box\* nex) {

next = nex;

}

Box\* Box::Get\_next() {

if (next) return next;

else return nullptr;

}

## Приложение 2. Листинг Section\_func.cpp

#include"Sklad.h"

Section::Section() {

head = nullptr;

tail = nullptr;

section\_id = -1;

count = 0;

}

Section::Section(int id) {

head = nullptr;

tail = nullptr;

section\_id = id;

count = 0;

}

void Section::Set\_section(int id) {

section\_id = id;

count = 0;

}

int Section::Get\_section\_id() {

return section\_id;

}

Box\* Section::Get\_section\_head() {

return head;

}

int Section::Get\_box\_count() {

return count;

}

bool Section::Is\_Empty() {

return count == 0;

}

void Section::Add\_box(int box\_id, bool box\_occupancy) {

if (Find\_box(box\_id) == -1) {

Box\* new\_tail = new Box (box\_id, box\_occupancy);

if (count == 0) {

head = tail = new\_tail;

}

else {

tail->Set\_next(new\_tail);

tail = new\_tail;

}

count++;

}

else {

std::cout << "Ячейка с таким номером уже сущетсвует \n";

}

}

void Section::Delete\_box() {

if (Is\_Empty()) {

std::cout << "Секция пустая. Удаление невозможно. \n";

}

else {

Box\* temp = head;

head = head->Get\_next();

free(temp);

count--;

}

}

void Section::Section\_status() {

if (!Is\_Empty()) {

Box\* temp = head;

std::cout <<"Всего ячеек в секции: " << count << "\n";

for (int i = 1; i < count + 1; i++) {

std::cout << " ячейка № " << temp->Get\_id() << " Состояние = ";

if (temp->Get\_occupancy()) std::cout << "заполнена \n";

else std::cout << "не заполнена \n";

temp = temp->Get\_next();

}

}

else {

std::cout << "Секция пустая. \n";

}

}

int Section::Find\_box(int bx\_id) {

int ret = -1;

Box\* curr = head;

while (curr != nullptr) {

if (curr->Get\_id() == bx\_id) ret = bx\_id;

curr = curr->Get\_next();

}

return ret;

}

Box\* Section::Ret\_box(int bx\_id) {

Box\* ret = head;

Box\* curr = head;

while (curr != nullptr) {

if (curr->Get\_id() == bx\_id) ret = curr;

curr = curr->Get\_next();

}

return ret;

}

## Приложение 3. Листинг Stock\_func.cpp

#include"Sklad.h"

Stock::Stock() {

count = 0;

}

void Stock::Set\_name(std::string n) {

name = n;

}

bool Stock::Is\_Empty() {

return count == 0;

}

bool Stock::Is\_Full() {

return count == MAX;

}

void Stock::Add\_section(int section\_id) {

if (Is\_Full()) {

std::cout << "Склад заполнен. Добавление невозможно. \n";

}

if (!Is\_Full() && !Is\_Empty()) {

if (Find\_section(section\_id) != -1) {

std::cout << "Секция с заданным номером уже существует \n";

}

else {

Section s = Section(section\_id);

sect[count] = s;

count++;

}

}

if (Is\_Empty()) {

Section s = Section(section\_id);

sect[0] = s;

count++;

}

}

int Stock::Find\_section(int section\_id) {

int i = 0;

if (Is\_Empty()) i = -1;

else {

while (sect[i].Get\_section\_id() != section\_id && i != MAX) i++;

if (i == MAX) i = -1;

}

return i;

}

void Stock::Stock\_status() {

if (!Is\_Empty()) {

std::cout << "Наименование склада: " << name << "\n";

for (int i = 0; i < count; i++) {

std::cout << " id секции = " << sect[i].Get\_section\_id() << "\n";

sect[i].Section\_status();

}

}

else {

std::cout << "Склад " << name << " пуст. \n";

}

}

void Stock::Delete\_section(int section\_id) {

int num = Find\_section(section\_id);

if (num != -1) {

while (sect[num].Get\_section\_head() != nullptr) sect[num].Delete\_box();

for (int i = num; i < count; i++) sect[i] = sect[i + 1];

count--;

}

else {

std::cout << "Заданной секции нет. Удаление не возможно. \n";

}

}

Section\* Stock::Ret\_section(int section\_id) {

int i = -1;

if (!Is\_Empty()) {

while (sect[i].Get\_section\_id() != section\_id && i != MAX) i++;

}

return &sect[i];

}

void Stock::Delete\_stock() {

int num = 0;

while (count != 0) {

while (sect[num].Get\_section\_head() != nullptr) sect[num].Delete\_box();

count--;

num++;

}

}

void Stock::Stock\_status\_F(std::string filename) {

std::string line;

std::ifstream fl(filename);

if (fl.is\_open())

{

std::getline(fl, line);

while (true)

{

if (fl.eof()) break;

int num = 0;

std::string kod\_name = "";

while (line[num] != ':' && num != line.length() && !isdigit(line[num])) {

kod\_name += line[num];

num++;

}

num += 2;

if (kod\_name == "Stock name") {

while (num != line.length()) {

name += line[num];

num++;

}

}

else if (kod\_name == "Section numbers") {

while (num < line.length()) {

std::string sec\_n = "";

while (line[num] != ' ' && num != line.length()) {

sec\_n += line[num];

num++;

}

Add\_section(std::stoi(sec\_n));

num++;

}

}

else if (kod\_name == "Box numbers and status for ") {

std::string sec\_n = "";

num = 27;

while (line[num] != ':' && num != line.length()) {

sec\_n += line[num];

num++;

}

num += 2;

while (num < line.length()) {

std::string box\_n = "";

std::string box\_stat = "";

while (line[num] != '-' && num != line.length()) {

box\_n += line[num];

num++;

}

while (line[num] != ' ' && num != line.length()) {

box\_stat += line[num];

num++;

}

Section\* reter = Ret\_section(std::stoi(sec\_n));

reter->Add\_box(std::stoi(box\_n), std::stoi(box\_stat));

num++;

}

}

std::getline(fl, line);

}

}

else {

std::cerr << "Не удалось открыть файл ввода. \n";

}

fl.close();

}

void Stock::Stock\_printf(const std::string filename) {

std::string line;

std::ofstream fl(filename);

if (fl.is\_open())

{

fl << "Stock name: " << name << "\n";

fl << "Section count: " << count << "\n";

fl << "Section numbers: ";

for (int i = 0; i < count; i++) {

fl << sect[i].Get\_section\_id() << " ";

}

fl << "\n";

for (int i = 0; i < count; i++) {

fl << "Box count for " << sect[i].Get\_section\_id() << ": ";

fl << sect[i].Get\_box\_count() << "\n";

fl << "Box numbers and status for " << sect[i].Get\_section\_id() << ": ";

Box\* bb = sect[i].Get\_section\_head();

while(bb != nullptr) {

fl << bb->Get\_id() << "-" << bb->Get\_occupancy() << " ";

bb = bb->Get\_next();

}

fl << "\n";

}

}

else {

std::cerr << "Не удалось открыть файл вывода. \n";

}

fl.close();

}

## Приложение 4. Листинг Menu\_func.cpp

#include"Menu.h"

void Menu::Text\_menu() {

std::cout << "Выберете необходимое действие\n";

std::cout << "1. Вывести состояние склада\n";

std::cout << "2. Вывести состояние секции с заданным номером\n";

std::cout << "3. Вывести состояние ячейки с заданным номером и заданной секцией\n";

std::cout << "4. Добавить секцию\n";

std::cout << "5. Добавить ячейку в заданную секцию\n";

std::cout << "6. Удалить секцию\n";

std::cout << "7. Удалить ячейку в заданной секции\n";

std::cout << "8. Заполнить данными из файла\n";

std::cout << "9. Вывести текущее состояние в файл\n";

std::cout << "10. Удалить все данные о складе\n";

std::cout << "11. Добавить наименование склада\n";

std::cout << "0. Завершить работу программы\n";

std::cout << "Ввод: ";

}

void Menu::Print\_stock\_status(Stock\* st){

std::cout << "Текущее состояние склада: \n";

st->Stock\_status();

}

void Menu::Print\_section\_status(Stock\* st) {

std::cout << "Введите номер секции \n";

int sect\_id = Input\_for\_menu(0);

if (st->Find\_section(sect\_id) != -1) {

Section\* sec = st->Ret\_section(sect\_id);

std::cout << "Текущее состояние указанной секции: \n";

sec->Section\_status();

}

else {

std::cout << "Секции с заданным номером не существует \n";

}

}

void Menu::Print\_box\_status(Stock\* st) {

std::cout << "Введите номер секции \n";

int sect\_id = Input\_for\_menu(0);

if (st->Find\_section(sect\_id) != -1) {

Section\* sec = st->Ret\_section(sect\_id);

std::cout << "Введите номер ячейки\n";

int bx\_id = Input\_for\_menu(0);

if (sec->Find\_box(bx\_id) != 0) {

Box\* bx = sec->Ret\_box(bx\_id);

std::cout << "Текущее состояние указанной ячейки: \n";

if (bx->Get\_occupancy()) std::cout << "Ячейка заполнена \n";

else std::cout << "Ячейка не заполнена \n";

}

else {

std::cout << "Ячейки с заданным номером не существует \n";

}

}

else {

std::cout << "Секции с заданным номером не существует \n";

}

}

void Menu::M\_add\_sect(Stock\* st) {

std::cout << "Введите номер секции, которую необходимо добавить: \n";

int new\_id = Input\_for\_menu(0);

st->Add\_section(new\_id);

}

void Menu::M\_add\_box(Stock\* st) {

std::cout << "Введите номер секции, куда необходимо добавить ячейку: \n";

int sect\_id = Input\_for\_menu(0);

if (st->Find\_section(sect\_id) != -1) {

Section\* sec = st->Ret\_section(sect\_id);

std::cout << "Введите номер ячейки, которую неоходимо добавить: \n";

int bx\_id = Input\_for\_menu(0);

std::cout << "Введите 1, если ячейка заполнена, и 0, если ячейка не заполнена: \n";

bool bx\_oc = Input\_for\_menu(2);

sec->Add\_box(bx\_id, bx\_oc);

}

else {

std::cout << "Секции с заданным номером не существует \n";

}

}

void Menu::M\_delete\_sect(Stock\* st) {

std::cout << "Введите номер секции, котрую необходимо удалить: \n";

int sect\_id = Input\_for\_menu(0);

st->Delete\_section(sect\_id);

}

void Menu::M\_delete\_box(Stock\* st) {

std::cout << "Введите номер секции, из которой необходимо удалить ячейку: \n";

int sect\_id = Input\_for\_menu(0);

if (st->Find\_section(sect\_id) != -1) {

Section\* sec = st->Ret\_section(sect\_id);

sec->Delete\_box();

}

else {

std::cout << "Секции с заданным номером не существует \n";

}

}

void Menu::Start(){

Stock st = Stock();

while (1) {

Text\_menu();

int ch = Input\_for\_menu(1);

if (ch == 1) Print\_stock\_status(&st);

else if (ch == 2) Print\_section\_status(&st);

else if (ch == 3) Print\_box\_status(&st);

else if (ch == 4) M\_add\_sect(&st);

else if (ch == 5) M\_add\_box(&st);

else if (ch == 6) M\_delete\_sect(&st);

else if (ch == 7) M\_delete\_box(&st);

else if (ch == 8) M\_Fadd\_stock(&st);

else if (ch == 9) M\_Fprint\_status(&st);

else if (ch == 10) M\_destroy(&st);

else if (ch == 11) M\_add\_name(&st);

else if (ch == 0) break;

}

}

void Menu::M\_add\_name(Stock\* st) {

std::string nm;

std::cout << "Введите наименование склада: " << std::endl;

std::cin >> nm;

st->Set\_name(nm);

}

void Menu::M\_destroy(Stock\* st) {

st->Delete\_stock();

}

void Menu::M\_Fprint\_status(Stock\* st) {

std::string f\_name;

std::cout << "Введите название файла: " << std::endl;

std::cin >> f\_name;

st->Stock\_printf(f\_name);

}

void Menu::M\_Fadd\_stock(Stock\* st) {

std::string f\_name;

std::cout << "Введите название файла: " << std::endl;

std::cin >> f\_name;

st->Stock\_status\_F(f\_name);

}

int Menu::Input\_for\_menu(int stat) {

int choice;

while (!(std::cin >> choice))

{

std::cin.clear();

while (std::cin.get() != '\n') {

std::cout << "Повторите ввод :" << std::endl;

}

}

if (stat == 1 && (choice > 11 || choice < 0)) {

std::cout << "Повторите ввод :" << std::endl;

choice = Input\_for\_menu(1);

}

if (stat == 2 && choice != 1 && choice != 0) {

std::cout << "Повторите ввод :" << std::endl;

choice =Input\_for\_menu(2);

}

if (stat == 3 && (choice <= 0 || choice > 10000)) {

std::cout << "Повторите ввод :" << std::endl;

choice = Input\_for\_menu(3);

}

return choice;

}

## Приложение 5. Листинг Kursach.cpp

#include"Menu.h"

int main()

{

setlocale(LC\_ALL, "rus");

Menu m = Menu();

m.Start();

return 0;

}

## Приложение 6. Листинг Menu.h

#pragma once

#include"Sklad.h"

class Menu {

public:

void Text\_menu();

void Print\_stock\_status(Stock\* st);

void Print\_section\_status(Stock\* st);

void Print\_box\_status(Stock\* st);

void M\_add\_sect(Stock\* st);

void M\_add\_box(Stock\* st);

int Input\_for\_menu(int stat);

void M\_delete\_sect(Stock\* st);

void M\_delete\_box(Stock\* st);

void M\_Fprint\_status(Stock\* st);

void M\_Fadd\_stock(Stock\* st);

void M\_destroy(Stock\* st);

void M\_add\_name(Stock\* st);

void Start();

};

## Приложение 7. Листинг Sklad.h

#pragma once

#include <iostream>

#include <fstream>

#include <stdlib.h>

#include <string>

#define MAX 10

class Box {

private:

int box\_id;

bool occupancy;

Box\* next;

public:

Box();

Box(int id, bool oc);

void Set\_id(int id);

int Get\_id();

void Set\_occupancy(bool occupancy);

bool Get\_occupancy();

void Set\_next(Box\* nex);

Box\* Get\_next();

};

class Section{

private:

int section\_id;

Box\* head;

Box\* tail;

int count;

public:

Section();

Section(int id);

void Set\_section(int id);

int Get\_section\_id();

int Get\_box\_count();

bool Is\_Empty();

void Add\_box(int box\_id, bool box\_occupancy);

void Delete\_box();

void Section\_status();

int Find\_box(int bx\_id);

Box\* Ret\_box(int bx\_id);

Box\* Get\_section\_head();

};

class Stock {

private:

std::string name;

Section sect[MAX]{};

int count;

public:

Stock();

bool Is\_Empty();

bool Is\_Full();

void Add\_section(int section\_id);

int Find\_section(int section\_id);

void Set\_name(std::string n);

void Stock\_status\_F(const std::string filename);

void Stock\_printf(const std::string filename);

void Delete\_stock();

void Stock\_status();

void Delete\_section(int section\_id);

Section\* Ret\_section(int section\_id);

};