Документация
Тема 3: Склад

Глава 1. Увод

Целта на проекта е програма, реализираща информационна система, обслужваща склад. Програмата съхранява и обработва данните за наличността в склада в текстов файл като работата с програмата се осъществява в диалогов режим.

От така формулираната цел произлизат следните задачи:

- Дефиниране на изисквания към системата.
- Конструиране и реализация на представянето на продуктите в склада.
- Конструиране и реализация на съставните елементи на склада – рафт, секция.
- Конструиране и реализация на склада и информационната система, която го обслужва.

Те ще бъдат разгледани в:

- Глава 2. Преглед на предметната област
- Глава 3. Проектиране
- Глава 4. Реализация

Глава 2. Преглед на предметната област

Системата трябва да поддържа операциите:

- Извеждане на информация за наличните продукти в склада.
- Добавяне на продукт в диалогов режим.
- Изваждане на продукт в диалогов режим.
- Извеждане на справка за всички промени в наличността в период, въведен от потребителя.
- Разчистване на склада от негодните продукти.

Глава 3. Проектиране

Проектът включва класовете:

- Constants – съдържа основните константи, използвани от останалите класове.
- Date – грижи се за обработката на дати.
- Product – конструиране и реализация на представянето на продукт в склада.
- Shelf – конструиране и реализация на рафт от склада, посредством саморазширяващ се масив от указатели към обекти на класа Product.
- Section – конструиране и реализация на секция от склада, посредством саморазширяващ се масив от указатели към обекти на класа Shelf.
- Warehouse – конструиране и реализация на склад, посредством саморазширяващ се масив от указатели към обекти на класа Section.
- InformationSystem – осъществява диалоговия режим на информационната система, обслужваща склад. Съдържа композиция на обект на класа Warehouse.
- Examples – съдържа примери за работата на отделните класове.

Глава 4. Реализация

- Клас **Constants**

Съдържа статични член-данни, със спецификатор за достъп public:

`static const size_t MAX_NAME_LEN = 255;` - максимална дължина на името на продукта.

`static const size_t MAX_MAKER_LEN = 255;` - максимална дължина на името на производителя на продукта.

`static const size_t MAX_COMMENT_LEN = 10000;` - максимална дължина на коментара на продукта.

`static const size_t INITIAL_CAPACITY = 2;` - начален капацитет на масивите от указатели.

`static const size_t INCREASE_STEP = 2;` - при преоразмеряване удвояваме капацитета на масивите от указатели.

`static const size_t MAX_SHELF_CAPACITY = 16;` - максимален брой деления на един рафт.

`static const size_t MAX_QUANTITY_IN_ONE_SHELF_DIVISION = 9;` - максимален брой продукти в едно деление на рафт от склада.

`static const size_t MAX_SECTION_CAPACITY = 16;` - максимален брой рафтове в една секция.

`static const size_t MAX_BUFFER_LEN = 10000;`

`static const size_t MAX_OPERATION_LEN = 20;` - максимален брой символи на операциите в класа InformationSystem.

- Клас **Date**

Съдържа 3 член-данни от тип unsigned int – day, month, year - със спецификатор за достъп private (няма пряк външен достъп до тях). Обектите на класа Date се създават посредством конструктор с параметри (от тип const unsigned int), които имат и стойности по подразбиране. За четене на стойностите на член-данните се използват селектори (член-функции от първо ниво), с тип на връщаната стойност unsigned int:

```
unsigned int getDay() const;
unsigned int getMonth() const;
unsigned int getYear() const;
```

За промяна на стойностите на член-данните на обект от този клас, се използва мутатор (член-функция от първо ниво), с тип на параметрите const unsigned int, грижещ се за валидация на данните данните преди окончателната промяна на стойностите:

```
void setDate(const unsigned int day, const unsigned int month, const unsigned int year);
```

Предефинирани са операции за сравняване на дати, приемащи константна препратка към обект от класа Date и булев тип на връщаната стойност:

```
bool operator == (const Date& other) const;
bool operator < (const Date& other) const;
```

Предефинирана е операция +=, добавяща до 28 дни към текущия обект, с тип на аргумента const int и тип на връщаната стойност препратка към обект на класа Date:

```
Date& operator += (const int days);
```
Използва се при разчистването на склада от продуктите с изтекъл или скоро изтичащ срок на годност.

Предефинирани са операции за вход (>>) и изход (<<) като приятелски функции:

```
friend std::ostream& operator << (std::ostream& out, const Date& date);
friend std::istream& operator >> (std::istream& in, Date& date);
```

Функцията за проверка дали една година е високосна, приема параметър от тип unsigned int и връща булев резултат, дефинирана е като статична функция, тъй като няма нужда от използване на член-данни или член-функции на текущия обект:

```
static bool isLeapYear(unsigned int year);
```

- Клас **Product**

Член-данните са със спецификатор за достъп private (няма пряк външен достъп до тях):

`char* name;` - име на продукта (динамично заделена член-данна с максимален размер 254 символа).

`Date expiryDate;` - срок на годност (композиция на обект от класа Date).

`Date dateOfEntry;` - дата на постъпване в склада (композиция на обект от класа Date).

`char* maker;` - име на производител (динамично заделена член-данна с максимален размер 254 символа).

`size_t quantity;` - налично количество.

Член-данни, пазещи местоположението в склада (номерирането започва от 1):

`unsigned int section;` - номер на секция .

`unsigned int shelf;` - номер на рафт (приема стойности между 1 и MAX_SECTION_CAPACITY).

`unsigned int number;` - пореден номер на деление от рафта (приема стойности между 1 и MAX_SHELF_CAPACITY) – едно деление се състои от няколко продукта, като количеството им се пази в константа от класа Constants - MAX_QUANTITY_IN_ONE_SHELF_DIVISION – предназначението й е свързано с това, че в различните складове могат да се пазят стоки, които заемат различен обем място.

`char* comment;` - коментар (динамично заделена член-данна с максимален размер 9999 символа).

Създаването на обекти на класа Product се осъществява чрез конструктор по подразбиране и конструктор с параметри. Заради използването на динамична памет за част от член-данните са реализирани конструктор за копиране, деструктор и операция =.

Селектори (член-функции от първо ниво) за четене на стойността на някои член-данни:

```
const char* getName() const;
Date getExpiryDate() const;
Date getDateOfEntry() const;
const char* getMaker() const;
size_t getQuantity() const;
```

Мутатори, грижещи се за коректната промяна на стойностите на член-данните (със спецификатор за достъп private):

```
void setName(const char* name);
void setExpiryDate(const Date& expiryDate);
```

```
void setDateOfEntry(const Date& dateOfEntry);
void setMaker(const char* maker);
void setComment(const char* comment);
```

Мутатори, грижещи се за коректната промяна на стойностите на член-данните (със спецификатор за достъп public):

```
void setQuantity(const int quantity)
void setSection(const int section);
void setShelf(const int shelf);
void setNumber(const int number);
```

Предефинирана е операция за сравняване (==) на два продукта - два продукта са еднакви, ако името на продукта и името на производителя съвпадат:

```
bool operator == (const Product& other) const;
```

Предефинирани са операции за вход (>>) и изход (<<) като приятелски функции:

```
friend std::ostream& operator << (std::ostream& out, const Product& p);
   friend std::istream& operator >> (std::istream& in, Product& p);
```

- Клас **Shelf**

Член-данните са със спецификатор за достъп private (няма пряк външен достъп до тях):

`Product** products;` - масив от указатели към обекти на класа Product (динамично заделен).

`size_t size;` - брой запълнени деления на рафта.

`size_t capacity;` - капацитет на масива products.

Обектите на класа Shelf се създават посредством конструктор по подразбиране. Заради динамично заделената член-данна products, за коректната обработка на паметта, са реализирани конструктор за копиране, деструктор и операция =.

Член-функции със спецификатор за достъп private (няма пряк външен достъп до тях):

`void allocate(const size_t size);` - помощна функция, заделя памет за size на брой указателя към обекти на класа Product.

`void copy(const Shelf& other);` - помощна функция за копиране на обект на класа Shelf в текущия обект.

`void deallocate();` - помощна функция за освобождаване на заделена памет за текущия обект.

`void setSize(const int size);` - мутатор за промяна на член-данната size.

`void setCapacity(const int capacity);` - мутатор за промяна на член-данната capacity.

`void resize();` - помощна функция за преоразмеряване на член-данната products.

`bool addProduct(const Product& product, const int index);` - помощна функция за добавяне на нов елемент в член-данната products на позиция index.

`void relocationInAscOrderOfIndices(const int from, const int to);` - помощна функция за изместване на елементите на член-данната products с 1 позиция.

`void relocationInDescOrderOfIndices(const int from, const int to);` - помощна функция за изместване на елементите на член-данната products с 1 позиция.

`void deleteProduct(const int index);` - помощна функция за изтриване на елемента на член-даннта products на позиция index.

Член-функции със спецификатор за достъп public:

`size_t getSize() const;` - селектор за броя елементи на член-данната products.

`bool removeProduct(const char* productName, const int quantity, int*& numbers, int& size, std::ostream& out = std::cout);` - премахва продукта с име productName и количество quantity, като извежда информацията за този продукт и връща булев резултат дали продуктът е премахнат успешно. При недостиг на количество, ще се задели памет за numbers, съхраняващ позициите, на които се среща този продукт на рафта, в size ще се пази техният брой и ще се върне стойност false. Ако премахването е успешно numbers = nullptr, size = 0 и ще се върне стойност true.

`size_t totalQuantityOfProduct(const Product& product) const;` - връща сумарното количество на product в текущия обект.

Предефинирана е операция +=, добавяща обект на класа Product към текущия обект:

```
Shelf& operator += (const Product& product);
```

Предефинирани са операции за индексиране:

```
Product& operator [] (int index);
const Product operator [] (int index) const;
```

Предефинирани са операции за вход (>>) и изход (<<) като приятелски функции:

```
friend std::ostream& operator << (std::ostream& out, const Shelf& shelf);
friend std::istream& operator >> (std::istream& in, Shelf& shelf);
```

- Клас **Section**

Член-данните са със спецификатор за достъп private (няма пряк външен достъп до тях):

`Shelf** shelves;` - масив от указатели към обекти на каса Shelf (динамично заделен).

`size_t size;` - брой добавени обекти на класа Shelf към текущия обект.

`size_t capacity;` - капацитет на масива shelves.

Обектите на класа Section се създават посредством конструктор по подразбиране. Заради динамично заделената член-данна shelves, за коректната обработка на паметта са реализирани конструктор за копиране, деструктор и операция =.

Член-функции със спецификатор за достъп private (няма пряк външен достъп до тях):

`void allocate(const size_t size);` - помощна функция, заделя памет за size на брой указателя към обекти на класа Shelf.

`void copy(const Section& other);` - помощна функция за копиране на обект на класа Section в текущия обект.

`void deallocate();` - помощна функция за освобождаване на заделена памет за текущия обект.

`void setSize(const int size);` - мутатор за промяна на член-данната size.

`void setCapacity(const int capacity);` - мутатор за промяна на член-данната capacity.

`void resize();` - помощна функция за преоразмеряване на член-данната shelves.

`bool hasTheSameProduct(const Product& product, const int shelfNumber);` - проверява дали на елемента от shelves с индекс shelfNumber има продукт, съвпадащ с product и неговия срок на годност.

Член-функции със спецификатор за достъп public:

`size_t getSize() const;` - селектор за броя елементи на член-данната shelves.

`void addProduct(const Product& product);` - добавя продукт към текущия обект.

`bool removeProduct(const char* productName, const int quantity, int**& locations, int& size, std::ostream& out = std::cout);` - премахва продукта с име productName и количество quantity, като извежда информацията за този продукт и връща булев резултат дали продуктът е премахнат успешно. При недостиг на количество в секцията, ще се задели памет за locations, съхраняващ номер на рафт и позициите, на които се среща този продукт на този рафт, в size ще се пази броят на тези двойки и ще се върне стойност false. Ако премахването е успешно locations = nullptr, size = 0 и ще се върне стойност true.

`size_t totalQuantityOfProduct(const Product& product) const;` - връща сумарното количество на product в текущата секция.

Предефинирана е операция +=, добавяща обект на класа Shelf към текущия обект:

```
Section& operator += (const Shelf& shelf);
```

Предефинирани са операции за индексиране:

```
Shelf & operator [] (int index);
const Shelf operator [] (int index) const;
```

Предефинирани са операции за вход (>>) и изход (<<) като приятелски функции:

```
friend std::ostream& operator << (std::ostream& out, const Section& section);
    friend std::istream& operator >> (std::istream& in, Section& section);
```

- Клас **Warehouse**

Член-данните са със спецификатор за достъп private (няма пряк външен достъп до тях):

`Section** sections;` - масив от указатели към обекти на каса Section (динамично заделен)

`size_t size;` - брой добавени обекти на класа Section към текущия обект.

`size_t capacity;` - капацитет на масива sections.

Обектите на класа Warehouse се създават посредством конструктор по подразбиране. Заради динамично заделената член-данна sections, за коректната обработка на паметта са реализирани конструктор за копиране, деструктор и операция =.

Член-функции със спецификатор за достъп private (няма пряк външен достъп до тях):

`void allocate(const size_t size);` - помощна функция, заделя памет за size на брой указателя към обекти на класа Section.

`void copy(const Warehouse& other);` - помощна функция за копиране на обект на класа Warehouse в текущия обект.

`void deallocate();` - помощна функция за освобождаване на заделена памет за текущия обект.

`void setSize(const int size);` - мутатор за промяна на член-данната size.

`void setCapacity(const int capacity);` - мутатор за промяна на член-данната capacity.

`void resize();` - помощна функция за преоразмеряване на член-данната sections.

`static bool allocateLocations(int**& locations, const int size);` - заделяне на памет за locations, в случай на недостатъчно количество от даден продукт в склада, пазещ тройките номер на секция, номер на рафт и позиция на рафта.

`static void deallocateLocations(int**& locations, const int size);` - освобождава заделената памет за locations.

Член-функции със спецификатор за достъп public:

`size_t getSize() const;` - селектор за броя елементи на член-данната shelves.

`bool addProduct(const Product& product);` - добавя продукт към текущия обект.

`bool removeProduct(const char* productName, const int quantity, int**& locations, int& _size, std::ostream& out = std::cout);` - премахва продукта с име productName и количество quantity, като извежда информацията за този продукт и връща булев резултат дали продуктът е премахнат успешно. При недостиг на количество в склада, ще се задели памет за locations, съхраняващ номер на секция, номер на рафт и позициите, на които се среща този продукт на този рафт, в _size ще се пази броят на тези тройките и ще се върне стойност false. Ако премахването е успешно locations = nullptr, _size = 0 и ще се върне стойност true.

`size_t totalQuantityOfProduct(const Product& product) const;` - връща сумарното количество на product в текущия склад.

Предефинирана е операция +=, добавяща обект на класа Section към текущия обект:

```
Warehouse& operator += (const Section& section);
```

Предефинирани са операции за индексиране:

```
Section& operator [] (int index);
const Section operator [] (int index) const;
```

Предефинирани са операции за вход (>>) и изход (<<) като приятелски функции:

```
friend std::ostream& operator << (std::ostream& out, const Warehouse& warehouse);
friend std::istream& operator >> (std::istream& in, Warehouse& warehouse);
```

- Клас **InformationSystem**

Реализиран на принципа на обектно-ориентирания шаблон за дизайн Singleton, притежава една инстанция и не могат да се създават обекти от него, както и да се копират такива, затова конструкторът за копиране и операцията за присвояване са изтрити.

Класът реализира информационната система, обслужваща склада в диалогов режим, съдържа една член-данна със спецификатор за достъп private (няма пряк външен достъп) – композиция на обект от клас Warehouse.

Член-функции:

`static void deallocateLocations(int**& locations, const int size);` - освобождава заделената памет за locations.

`static char* processFileName(int day, int month, int year);` - заделя динамично памет за името на файла, генериран при разчистването на склада от негодни продукти и го обработва, за да е във формата зададен в условието.

`static size_t getFileSize(const char* fileName);` - намира размера на файл с име filename.

`static void clearFileWithChanges();` - изтрива съдържанието на файла "changes.txt", съхраняващ промените в наличността на склада.

`void availability() const;` - извежда наличните продукти в склада.

`void addProduct();` - добавя продукт в склада в диалогов режим.

`void removeProduct();` - премахва продукт от склада в диалогов режим.

`static void referenceForChanges();` - извежда справка за всички промени в наличността в период, въведен от потребителя.

`void clear();` - разчиства склада от всички стоки, на които е изтекъл или предстои скоро да изтече срока на годност.

`void useInformationSystem();` - осъществява и поддържа диалоговия режим.

- Клас **Examples –** съдържа статични член-функции, съдържащи примери за работата на отделните класове.

Глава 5. Заключение

Разработената програма симулира информационна система, обслужваща склад, съхранява и обработва данните за наличността в склада в текстов файл. Вероятно може да се постигне подобрение в архитектурата.

Изготвил: Николета Тонева Щерева