Magazyn

2016

Spis treści

1 Podstawowe informacje

Język: C++. Punkty:

Celem tego zadania będzie napisanie uproszczonej części systemu inwentaryzacji magazynu. Do napisania będzie kilka klas, bez funkcji `main`. Należy wysłać archiwum **zip**. W archiwum, w głównym katalogu, musi być plik `Warehouse.h` z definicją odpowiednich klas.

W Magazynie (*Warehouse*) przechowywane są Przedmioty (*Item*), każdy Przedmiot ma nazwę (i tylko nazwę, dla uproszczenia). Przedmioty przechowywane są w Sekcjach (*Section*) na numerowanych półkach. Sekcje zestawione są w Alejki (*Aisle*). Układ alejek tworzy cały magazyn. Przedmiot będący już w Magazynie (*WarehouseItem*) składa się z określenia przedmiotu oraz z jego położenia (*Location*), na które składają się Magazyn, Alejka i Sekcja dane jako odnośniki do odpowiednich obiektów oraz numer półki.

Zasadniczo układ magazynu (alejki, sekcje) tworzony jest raz i nie jest modyfikowany, ale można dodawać i usuwać przedmioty. Przedmioty można też wyszukiwać po początku nazwy.

W rozdziale 3 znajduje się wykaz klas oraz ich publicznych składowych (metod i konstruktorów), które należy dostarczyć. Oprócz składowych z wykazu potrzebne są publiczne **destruktory** - trzeba je napisać tam, gdzie domyślne nie wystarczają. Klasy powinny mieć też poprawne konstruktory kopiujące, chyba że zaznaczono inaczej. Oczywiście, jeżeli konstruktor domyślnie wytworzony działa dobrze, to nie trzeba go pisać. Klasa `Warehouse` nie musi być kopiowalna.

Poza składowymi wymienionymi poniżej oraz destruktorami i ew. konstruktorami kopiującymi wspomnianymi powyżej klasy nie powinny mieć żadnych publicznych ani chronionych składowych (zmiennych, pól, konstruktorów itd.). W niektórych przypadkach potrzebne może być użycie słowa **friend**. Zadania nie spełniające tego warunku mogą zostać odrzucone (lub ich punktacja obniżona) nawet po przejściu testów BaCy.

2 Uwagi

Wydajność nie jest głównym przedmiotem testów. Można przy każdym dodawaniu i usuwaniu przedmiotów przepisać wszystkie przedmioty z jednej sekcji (ale nie z całego magazynu). Natomiast różnice w czasie wykonania programów mogą być dość duże w zależności od użytych algorytmów. Za bardzo szybki program nie będzie dodatkowych punktów, co najwyżej “prawo do przechwałek”. Trzeba natomiast w miarę rozsądnie gospodarować pamięcią.

Dozwolone nagłówki to: `#include<string>` i ew. `#include <cstring>`. Proszę też nie używać wyrażeń podobnych do `#include` w komentarzach ani w ogóle nigdzie w załączonych plikach.

W wysłanym archiwum może znaleźć się plik `Warehouse.cpp` z definicjami metod poza zakresem klasy. Jeżeli ktoś nie wie, o czym piszę i jak z tego skorzystać, proszę umieścić wszystko w `Warehouse.h`.

3 Dokumentacja klas

3.1 Dokumentacja klasy Aisle

Klasa reprezentująca alejkę. Ma numer, przechowuje sekcje (Section).

Metody publiczne

- Aisle (int number, int sectionsNumber, Section **sections)
- Aisle (const Aisle &other)
- int getSectionsNumber () const
- Section *const * getSections () const
- int getNumber () const

3.1.1 Dokumentacja konstruktora i destruktora

3.1.1.1 Aisle::Aisle (int *number,* int *sectionsNumber,* Section ** *sections*)

Parametry

number	numer alejki
sectionsNumber	liczba sekcji w tablicy podanej jako kolejny parameter
sections	tablica wskaźników do sekcji (Section); ani tablica, ani tym bardziej sekcje nie są kopiowane, tylko przypisywany jest wskaźnik; alejka przejmuje zarządzanie pamięcią

3.1.1.2 Aisle::Aisle (const Aisle & *other*)

Konstruktor kopiujący "głęboko". W szczególności powinny zostać wykonane kopie wszystkich sekcji (Section::-Section(const Section &)) (ale nie przedmiotów w nich).

3.1.2 Dokumentacja funkcji składowych

3.1.2.1 int Aisle::getNumber ( ) const

Zwraca numer sekcji w alejce.

3.1.2.2 Section* const* Aisle::getSections ( ) const

Zwraca sekcje w kolejności numeracji. Zarządzanie pamięcią pozostaje w gestii alejki.

3.1.2.3 int Aisle::getSectionsNumber ( ) const

Zwraca liczbę sekcji.

3.2 Dokumentacja klasy Item

Reprezentacja przedmiotu. Dla uproszczenia przechowuje tylko nazwę.

Metody publiczne

- string **getName** () const
- **Item** (const string &name)

3.3 Dokumentacja klasy Warehouse::ItemSequence

Sekwencja przedmiotów wykorzystywana do zwracania wyników wyszukiwania (Warehouse::search()). Nie powinna zawierać kopii przedmiotów, tylko wskaźniki do tych już istniejących.

Metody publiczne

- ItemSequence (const ItemSequence &other)
- void reset ()
- WarehouseItem * next ()

3.3.1 Dokumentacja konstruktora i destruktora

3.3.1.1 Warehouse::ItemSequence::ItemSequence (const ItemSequence & *other*)

Kopiuje sekwencję - zarówno wskaźniki do przedmiotów jak i pozycję, na której jest ustawiona. Kopię można jednak niezależnie iterować.

3.3.2 Dokumentacja funkcji składowych

3.3.2.1 WarehouseItem* Warehouse::ItemSequence::next ( )

Zwraca następny przedmiot. Pierwsze wywołanie zwraca pierwszy. NULL jeśli nie ma więcej przedmiotów.

3.3.2.2 void Warehouse::ItemSequence::reset ( )

Ustawia sekwencję od pierwszego przedmiotu.

3.4 Dokumentacja klasy Location

Położenie przedmiotu składa się ze wskaźników do zawierających go obiektów magazynu, alejki, sekcji oraz numeru półki.

Metody publiczne

- **Location** (const Warehouse &_warehouse, const Aisle &_aisle, const Section &_Section, int _shelf)
- const Aisle & **getAisle** () const
- const Section & **getSection** () const
- const Warehouse & **getWarehouse** () const
- int **getShelf** () const

3.5 Dokumentacja klasy Section

Klasa reprezentująca sekcję. Przechowuje przedmioty.

Metody publiczne

- Section (int number, int shelvesNumber)
- Section (const Section &other)
- int getShelvesNumber () const
- int getNumber () const
- WarehouseItem ∗const ∗ getItems () const
- int getItemsNumber () const

3.5.1 Dokumentacja konstruktora i destruktora

3.5.1.1 Section::Section (int *number,* int *shelvesNumber*)

Parametry

number	numer sekcji
shelvesNumber	liczba półek

3.5.1.2 Section::Section (const Section & *other*)

Konstruktor kopiujący. Kopia nie zawiera żadnych przedmiotów! Nie jest to zatem prawdziwa kopia (i niekoniecznie najlepsza praktyka - tu dana dla przećwiczenia).

3.5.2 Dokumentacja funkcji składowych

3.5.2.1 WarehouseItem∗ const∗ Section::getItems ( ) const

Zwraca przedmioty w sekcji w kolejności dodawania. Zarządzanie pamięcią pozostaje w gestii sekcji (na zwróconym wyniku nie jest wywoływane `delete[]`).

3.5.2.2 int Section::getItemsNumber ( ) const

Zwraca liczbę przedmiotów w sekcji.

3.5.2.3 int Section::getNumber ( ) const

Zwraca numer sekcji.

3.5.2.4 int Section::getShelvesNumber ( ) const

Zwraca liczbę półek.

3.6 Dokumentacja klasy Warehouse

Główna klasa magazynu.

Komponenty

- class ItemSequence

  Sekwencja przedmiotów wykorzystywana do zwracania wyników wyszukiwania (Warehouse::search()). Nie powinna zawierać kopii przedmiotów, tylko wskaźniki do tych już istniejących.

Metody publiczne

- Warehouse (const string &name, int aislesNumber, Aisle **aisles)
- string **getName** () const
- WarehouseItem * addItem (const Item &p, int aisleNr, int sectionNr, int shelfNr)
- void deleteItem (WarehouseItem *wItem)
- Aisle *const * getAisles () const
- int getAislesNumber () const
- ItemSequence search (const string &namePrefix) const

3.6.1 Dokumentacja konstruktora i destruktora

3.6.1.1 Warehouse::Warehouse (const string & *name*, int *aislesNumber*, Aisle ** *aisles*)

Parametry

name	
aislesNumber	liczba alejek podanych jako tablica w kolejnym parametrze
aisles	tablica wskaźników do alejek (Aisle); kopiowany jest tylko ostateczny wskaźnik, nie jest wykonywana głęboka kopia

3.6.2 Dokumentacja funkcji składowych

3.6.2.1 WarehouseItem* Warehouse::addItem (const Item & *p*, int *aisleNr*, int *sectionNr*, int *shelfNr*)

Dodaje nowy przedmiot WarehouseItem do odpowiedniej sekcji.

Parametry

p	Przedmiot, którego kopię będzie zawierał nowy WarehouseItem
aisleNr	numer alejki, w której będzie przedmiot
sectionNr	numer sekcji
shelfNr	numer półki

Zwraca nowoutworzony przedmiot.

3.6.2.2 void Warehouse::deleteItem (WarehouseItem * *wItem*)

Usuwa przedmiot z magazynu. Zwalnia pamięć przez niego zajmowaną. Wszystkie istniejące w tym momencie sekwencje (ItemSequence) wskazujące przedmioty z tej samej sekcji stają się nieprawidłowe (w testach nie będą używane).

Parametry

wItem	wskaźnik do przedmiotu z magazynu, musi to być wskaźnik do przedmiotu w magazynie (uzyskany z Section::getItems(), Warehouse::addItem(const Item & p, int aisleNr, int sectionNr, int shelfNr) albo Warehouse::search())

3.6.2.3 Aisle* const* Warehouse::getAisles ( ) const

Zwraca Alejki w magazynie ponumerowane od 1 do Warehouse::getAislesNumber().

3.6.2.4 int Warehouse::getAislesNumber ( ) const

Zwraca liczbę alejek w magazynie.

3.6.2.5 ItemSequence Warehouse::search (const string & *namePrefix*) const

Wyszukuje przedmioty po prefiksie nazwy. Wielkość liter jest istotna.

Parametry

prefix	prefiks nazwy przedmiotów wyszukiwanych

Zwraca sekwencję pasujących przedmiotów w kolejności rosnących numerów alejek, później sekcji, później kolejności dodawania.

3.7 Dokumentacja klasy WarehouseItem

Klasa łącząca przedmiot z jego lokalizacją. Przechowuje wewnętrzne kopie zarówno przedmiotu jak i lokalizacji.

Metody publiczne

- **WarehouseItem** (const Item &item, const Location &location)
- const Item & **getItem** () const
- const Location & **getLocation** () const

4 Przykład użycia

Dla kodu

```
#include "Warehouse.h"
#include <iostream>
#include <string>
using std::cout;
using std::endl;
void printLocation(const Location & loc){
  cout << loc.getWarehouse().getName()
     << "-" <<loc.getAisle().getNumber()
     << "-" << loc.getSection().getNumber()
     << "-" << loc.getShelf();
}
void printWarehouseItem(const WarehouseItem & item){
  cout << item.getItem().getName() << " ";
  printLocation(item.getLocation());
  cout << endl;
}
void printWarehouse(const Warehouse & war){
  cout << "warehouse " << war.getName() << ":" << endl;
  for (int i = 0; i < war.getAislesNumber(); i ++){
```

```
    Aisle * aisle = war.getAisles()[i];
    cout << "aisle " << aisle->getNumber() << ":" << endl;
    for (int j = 0; j < aisle->getSectionsNumber(); j++){
      Section * section = aisle->getSections()[j];
      cout << "    section " << section->getNumber() << ":" << endl;
      for (int k = 0; k < section->getItemsNumber(); k++){
        cout << "        ";
        printWarehouseItem(*section->getItems()[k]);
      }
    }
  }
}
int main(int argc, char** argv)
{
  //Tworzymy magazyn, o długościach alejek 4 5 i 6
  //pierwsze 4 sekcje z alejki mają 4 półki, pozostałe 3
  Section *** sections =  new Section**[3];
  Aisle ** aisles = new Aisle*[3];
  for (int i = 0; i < 3; i++){
    sections[i] = new Section*[i+4];
    for (int j = 0; j < i+4; j++){
      sections[i][j] = new Section(j+1, 3 + ((j < 4) ? 1 : 0));
    }
    aisles[i] = new Aisle(i+1, i+4, sections[i]);
  }

  delete [] sections;
  Warehouse war("W1", 3, aisles);

  //Dodajemy przedmioty
  printWarehouseItem(*war.addItem(Item("Tłumik 122 Pal."), 1, 1, 1));
  printWarehouseItem(*war.addItem(Item("Tłumik 151 Pal."), 1, 2, 1));
  printWarehouseItem(*war.addItem(Item("Resor 1218 Pal."), 2, 1, 1));
  printWarehouseItem(*war.addItem(Item("Resor 1219 Pal."), 3, 6, 3));
  printWarehouseItem(*war.addItem(Item("Amortyzator 1329 Skrz."), 1, 3, 2));
  printWarehouseItem(*war.addItem(Item("Amortyzator 1379 Skrz."), 2, 1, 1));

  //Wypisujemy
  printWarehouse(war);

  //Kopia alejek, przedmioty NIE są kopiowane
  Aisle aisle = (*war.getAisles()[1]);
  Section s (*aisle.getSections()[0]);

  cout << s.getItemsNumber() << endl;

  //Przykład wyszukiwania i iteracji po sekwencji przedmiotów.
  Warehouse::ItemSequence seq = war.search("Resor");
  WarehouseItem * it;
  while (  (it = seq.next()) != NULL ){
    printWarehouseItem(*it);
  }

  seq.reset();

  //Przykłady usuwania
  war.deleteItem(seq.next());
  war.deleteItem(war.getAisles()[0]->getSections()[0]->getItems()[0]);
```

```
  cout << endl;
  printWarehouse(war);

  return 0;
}
```

Wyjściem będzie:

```
Tłumik 122 Pal. W1-1-1-1
Tłumik 151 Pal. W1-1-2-1
Resor 1218 Pal. W1-2-1-1
Resor 1219 Pal. W1-3-6-3
Amortyzator 1329 Skrz. W1-1-3-2
Amortyzator 1379 Skrz. W1-2-1-1
warehouse W1:
aisle 1:
    section 1:
        Tłumik 122 Pal. W1-1-1-1
    section 2:
        Tłumik 151 Pal. W1-1-2-1
    section 3:
        Amortyzator 1329 Skrz. W1-1-3-2
    section 4:
aisle 2:
    section 1:
        Resor 1218 Pal. W1-2-1-1
        Amortyzator 1379 Skrz. W1-2-1-1
    section 2:
    section 3:
    section 4:
    section 5:
aisle 3:
    section 1:
    section 2:
    section 3:
    section 4:
    section 5:
    section 6:
        Resor 1219 Pal. W1-3-6-3
0
Resor 1218 Pal. W1-2-1-1
Resor 1219 Pal. W1-3-6-3

warehouse W1:
aisle 1:
    section 1:
    section 2:
        Tłumik 151 Pal. W1-1-2-1
    section 3:
        Amortyzator 1329 Skrz. W1-1-3-2
    section 4:
aisle 2:
    section 1:
        Amortyzator 1379 Skrz. W1-2-1-1
    section 2:
    section 3:
    section 4:
    section 5:
```

```
aisle 3:
    section 1:
    section 2:
    section 3:
    section 4:
    section 5:
    section 6:
        Resor 1219 Pal. W1-3-6-3
```