Kuriérska Služba

Adrián Pauer

January 22, 2023

Abstract

Projekt zachytáva prácu z triedami, prúdmi, výnimkami, čítaním/ukladním dát do súboru. Cieľom je pokus o malé umelecké dielo.

1 Trieda Address-úloha1

Trieda uchováva "poštové číslo" a krajinu.

```
Address::Address() {
// nastavi hodnoty postalCode a coutry na 0 a "UNDEFINED". }
```

```
std::string Address::removeSpaces(const std::string &word)const{
// ako vystup vrati slovo word bez medzier na zaciatku a na konci}
```

```
void Address::putAddress(const std::string &completeAdress){
// metoda dostane vstup v tvare "postalCode#country".
// Ak ma postalCode viac ako 5 cislic metoda vyhodi vynimku
// WrongInput("WRONG POSTAL CODE"). Ak completeAdress nema spravny tvar
// vyhodi WrongInput("Address has wrong format"), Ak cast country je prazdna
// alebo obsahuje iny znak ako v intervaloch ['a','z'] alebo ['A','Z'] metoda
// vyhodi vynimku WrongInput ("WRONG COUNTRY"),inak nastavi clenske premenne
// coutry a postalCode
```

```
std::string Address::getAddress()const {
// vrati adresu v tvare "postalCode#country"};
```

2 Trieda Package-úloha2

Trieda slúži na reprezentáciu zásielky. Uchováva váhu ,adresu a mesto kde má byť doručená daná zásielka.

```
Package(const std::string &package){
// dostane vstup ako "vaha'\n'adresa",ak je vaha aj adresa v poriadku,
// nastvi clenske premenne,
// inak nastavi address na prazdnu adersu a town na ""}
```

```
bool putPackage(const std::string &package ){
// vstup v tvare ako predosla metoda. Skontroluje nezapornost vahy a ak je adresa spravne
// zadana nastavi clenske premenne weight,address, premennu town nastavi pomocou setTown()
// a vrati true, inak vrati false}
```

```
void setTown(const std::string &postalCode){
// z postalCode si zobere 2 najvacsie cislice vypocita index (kluc) daneho
// mesta v mape mapOfTowns, kde index = (sucet 2 najvacsich cislic)%12
// a nastavi clensku premennu town}
```

```
bool operator < (const Package &other)const{
// porovna zasielky podla parametra weight}
```

```
bool operator > (const Package &other)const{
// porovna zasielky podla parametra weight}
```

3 Trieda CourierCar-úloha3

Trieda reprezentuje Kuriérske auto. Uchováva počet a celkovú hmotnosť zásielok, vektor zásielok, a mesto, kde sa kuriér nachádza.

```
bool putPackage(const std::string &package){
// !! nasa sluzba rozvaza iba slovenske zasielky !!
// ak je zasielka validna (spravna vaha,adresa) a celkova hmotnost
// nie je vacsia ako maxWeight vlozi zasielku do packages vrati true,inak vrati false}
```

```
int putPackages(const std::string &packages){
// string packages obsahuje niekolko zasielok, ktore su oddelene 1 prazdnym riadkom
// metoda postupne prida vsetky validne zasielky do clenskej premennej packages a
// vrati pocet pridanych zasielok}
```

```
void sortPackages(){
// metoda usporiada zasielky od najlahsej po najtazsiu}
```

```
Package &operator [] (int index) {
// vrati zasielku a danom indexe ak je index mimo rozsahu vrati prazdnu zasielku}
```

```
bool readFromFile(const std::string &filename){
// Ak subor nie je mozne otvorit alebo kurier uz packages naplnene metoda vrati false.
// pomocou metody getline nacita postupne vsetky riadky zo suboru.
// Kedze subor obsahuje zasielky oddelene prazdnym riadkom, ukladajte do stringu
// riadky 1 zasielky oddelene \n. Nasledne pridajte zasielku pomocou putPackage,
// takto az do konca suboru, nasledne metoda vrati true.}
```

```
bool saveToFile(const std::string &filename){
// Ak subor nie je mozne otvorit alebo je vector packages prazdny metoda vrati false
// Inak metoda postupne po riadkoch vlozi vsetky zasielky do suboru oddelene
// prazdnym riadkom, nasledne vrati true. }
```

4 ShippingArea-úloha4

Trieda slúži na reprezentáciu oblasti, v rámci ktorej rozvážame zásielky. Ak medzi dvoma mestami existuje v graphOfArea hrana, voláme ich susedné

```
void setLocationOfCar(const std::string &location1){
// Ak sa location1 nachadza vo vectorOfTowns naplni premennu locationOfCar
// Inak nerobi nic }
```

```
ShippingArea(const std::string &filename,const CourierCar &courier1){
// metoda sa pokusi vytvorit graf pomocou makeGraphFromFile, ak sa podari,
// naplni premenne courier, graphOfArea
// inak nastavi premennu courier na prazdnu instanciu courierCar }
```

```
bool makeGraphFromFile(const std::string &filename){
// Ak subor nie je mozne otvorit metoda vrati false
// Inak do premennej graphOfArea vytvori mapu map susednosti,
// tj. kazdemu mestu (kluc) prislucha mapa susednosti (hodnota),
// ktora obsahuje susedne mesta (kluce) a ich vzdialenosti (hodnoty).
// Subor je konstruovany tak, ze 1 riadok obsahuje 2 susedne mesta a vzdialenost medzi nimi.
// Format : MESTO1 MESTO2 VZDIALENOST
// Ked metoda precita vsetky riadky, vrati true}
```

```
std::string getNeighbours(const std::string &town1) const{
// Vrati string miest oddelenych ',', s ktorymi susedi town1}
```

```
int getDistance(const std::string &town1,const std::string &town2)const{
// vrati vzdialenost medzi mestami town1 a town2}
```

```
std::set<std::string> townsToVisit()const{
// vrati mnozinu miest, ktore kurier musi navstivit aby dorucil vsetky zasielky}
```

```
std::map<std::string,int> getMapOfneighbours(const std::string &town1)const{
// vrati mapu susednch miest(kluce) a vzdialenosti(hodonty) patriace ku mestu town1
// ak town1 neexistuje v graphOfArea, metoda vrati prazdnu mapu}
```

```
bool isSubset(std::set<std::string> &container1,std::vector<std::string> &container2) const{
// ak vsetky prvky z container1 su aj v container2 vrati true
// inak metoda vrati false}
```

```
std::string findPossiblePathForCourier()const{
// predpokladame, ze path v grafe existuje
// metoda si inicializuje premenne ako path, resulthPath, shippingTowns, visitedTowns,
// ktore nasledne posle backrackingovej metode backtracking(). Ked skonci backtracking()
// nasa metoda vrati 1 moznu postupnost ako string miest oddelenych ',' v poradi,
// v akom ich kurier bude prechadzat tak, aby navstivil 1 mesto prave raz
// a presiel vsetky mesta, do ktorych ma zaniest zasielku. Cesta zacina tam kde je kurier. }
```

```
void backtracking(std::string currentTown,std::set<std::string>&visited,
    std::vector<std::string> &path, std::set<std::string>&shippingTowns,
   std::vector<std::string> &resultPath)const{
// metoda prehlada vsetky moznosti pricom prehladava susendne mesta currentTown. Ked nejake
// navstivi prida ho do path, visited.
// Ked narazi na cestu, kde navstivi vsetky potrebne mesta, v premennej resultPath
// si ulozi danu cetu.}
```

5 ShippingArea-úloha5

```
std::pair<std::string,int> minimumCostPath(const std::string &town1,const std::string
    &town2) const{
/// predpokladame, ze path v grafe existuje
// metoda si inicializuje premenne ako path, resulthPath, shippingTowns, visitedTowns,
// distance, ktore nasledne posle backrackingovej metode minimumPathRecursion().
// Ked skonci minimumPathRecursion() v premennej resulthPath a distacne bude cesta a
// prejdena vzdialenost, kde cesta je najratsia mozna z town1 do town2 a
// obsahuje vsetky potrebne mesta. Kazde mesto sa v ceste vyskytuje prave raz.
// Metoda vrati st::pair, kde first je string miest oddelenych ',',ktore tvoria cestu a
// second prejdena vzdialenost}
```

```
void minimumPathRecursion(const std::string &currentTown,const std::string&finalTown,
              std::set<std::string>&visited,std::vector<std::string>&path,
              std::vector<std::string>&finalPath,const int &currentDistance,
              int &shortestDistance ) const{
// metoda funguje podobne ako backtracking(), az na to, ze teraz najdeme finalPath iba
// vtedy, ked sa rekurzia dostane do finalTown, path obsahuje vsetky potrebne mesta a
// aktualna vzdialenost je mensia ako najkratsia doteraz najdena.}
```

```
std::pair<int,std::string> courierMinimumCostPath() const{
// predpokladame ze path v grafe existuje
// metoda najde najkratsiu cestu, ktora navstivi vsetky potrebne mesta a zacina v meste,
// kde sa nachadza kurier. Mozne riesenie: Mesta, ktore treba navstivit su
// potencialne posledne mesta v hladanej ceste.
// Do setu si ulozim vsetky std::pair(vzdialenost,vhodna cesta). Nasledne vratim prvy pair}
```