Strukture podataka

Implementacija liste povezanim memorijskim lokacijama  
Laboratorijske vježbe 4

Tuzla, april/travanj 2020.

# Sadržaj

[**Sadržaj**](#_7xmovcvuls9g) **2**

[**Uvod**](#_6qnw6wpfd4f3) **3**

[**Zadatak 1**](#_obz0sbe92ccd) **3**

[**Zadatak 2**](#_vpgrvwbxnya9) **3**

[**Zadatak 3**](#_mtbk9bp1zea8) **3**

[**Zadatak 4**](#_94smnern9w5v) **4**

[**Zadatak 5**](#_ae16oq720xy9) **4**

[**Dodatno**](#_87p9pk7o25ss) **4**

# Uvod

Unutar file-a **node.hpp** nalazi se implementacija jednog čvora jednostruko povezane liste. Unutar file-a **forwardlist.hpp** nalazi se implementacija liste povezanim memorijskim lokacijama. Unutar ovog file-a potrebno je izvršiti naredne zadatke.

# Zadatak 1

# Zadatak 2

Implementirati metodu:

* void forward\_list<T>::**transform**(U unary\_op);

Koja kao ulazni parametar prima funkciju (unarna operacija potpisa void(T&)) koja se primjenjuje nad svakim čvorom liste.

Odrediti asimptotsku složenost ove metode.

# Zadatak 3

a) Implementirati metodu:

* forward\_list<T>::Iterator forward\_list<T>::**find**(U predicate)

Koja kao ulazni parametar prima predikat funkciju (metodu) koja definiše uslov (filter) na osnovu kojeg je potrebno pronaći element unutar liste i vratiti **Iterator** nađenog elemenata.

U slučaju da element ne postoji u listi funkcija vraća vratiti **Iterator** koji pokazuje na kraj liste.

Odrediti asimptotsku složenost ove metode.

b) Implementirati metodu:

* forward\_list<T>::Iterator forward\_list<T>::**find**(const T& value)

Koja kao ulazni parametar prima konstantu referencu na objekat koji želimo pronaći u listi. Funkcija vraća Iterator nađenog elementa. U slučaju da element ne postoji u listi funkcija vraća vratiti **Iterator** koji pokazuje na kraj liste.

Moguće koristiti funkciju implementiranu iz Zadatka 2.

Odrediti asimptotsku složenost ove metode.

# Zadatak 4

Implementirati metodu:

* void forward\_list<T>::**erase**(forward\_list<T>::Iterator position)

Koja kao ulazni parametar prima **Iterator** elementa kojeg želimo izbrisati iz liste.

Odrediti asimptotsku složenost ove metode.

# Zadatak 5

Implementirati metodu:

* void forward\_list<T>::**splice**(Iterator position, forward\_list<T>& otherList)

Koja spaja (uklapa) **otherList** na proslijeđenu poziciju unutar liste nad kojom je ovaj metoda i pozvan.

Primjer kako ovaj metod funkcioniše je sljedeći:

Trenutna lista: 1 2 3 4 5 6 7 8 9

Druga lista: 0 1 0 1

Pozicija: Iterator od elementa sa vrijednošću 3

Stanje liste nakon poziva splice metoda: 1 2 **0 1 0 1** 3 4 5 6 7 8 9  
Druga lista je prazna nakon poziva splice metoda.

Odrediti asimptotsku složenost ove metode.

# Dodatno

Unutar file-a **main.cpp** nalaze se pozivi metoda iz prethodno opisanih zadataka.

Nakon implementacije jednog od metoda odkomentarisati dio koda koji testira dati metod i provjeriti njegovu ispravnost.