Структури от данни
Домашна работа №1

Пояснение:

- Предадените от Вас решения трябва да могат да се компилират успешно на Visual C++.

Изисквания за предаване:

- Всички задачи ще бъдат проверени автоматично за преписване. Файловете с голямо съвпадение ще бъдат проверени ръчно и при установено плагиатство ще бъдат **анулирани**.

- Предаване на домашното в указания срок от всеки студент като един **.zip архив** със следното име:
  - (номер_на_домашно)_IS_(курс)_(група)_(факултетен_номер)
  - (номер_на_домашно) е цяло число, отговарящо на **номера на домашното**, за което се отнася решението (например 1);
  - (курс) е цяло число, отговарящо на **курса Ви** (например 2);
  - (група) е цяло число, отговарящо на **административната Ви група** (например 1);
  - (факултетен_номер) е низ, отговарящ на **факултетния Ви номер** (например 12345 или 1MI01234);

Пример за .zip архив на текущото домашно: 1_IS_2_1_12345.zip

Архивът да съдържа само изходен код (.cpp и .h/.hpp файлове) с решение, отговарящо на условията на задачите, като файловете с изходен код за всяка задача трябва да са разположени в папка с име (номер_на_задача).

Качването на архива става на посоченото място в Moodle.

Структури от данни
Домашна работа №1

Задача 1:

Трябва да се направи браузър. На вас ви се пада честта да направите **модула за табове на браузъра**.

Всеки таб има **URL** (символен низ) и **дата** и **време** на зареждане на страницата под формата на *unix timestamp* (цяло число). **URL**-ът е този на текущо заредената в таба страница. Когато дадена страница се зареди в таба, за нея автоматично се запомня *unix timestamp* с времето и датата на достъпване. За да вземете *timestamp* за текущото време, може да използвате вградена функция.

В даден момент потребителят на браузъра може да работи само с един от всички табове, който ще наричаме **текущ**. От текущия таб можем да се прехвърляме върху някой от непосредствените му съседи – или този преди него, или този след него. Разбира се, това става само ако има такива (*например, първият таб няма друг преди него*). За целта трябва да се поддържат операции за преминаване **назад** (*back*) или **напред** (*forward*) спрямо текущия таб.

Приемаме, че в браузъра винаги има отворен **поне един** таб. Ако е отворен само един таб и потребителят го затвори, то да се отвори нов таб, който е с адрес "*about:blank*". Когато браузърът се стартира, още преди потребителят да може да въвежда команди, в него веднага се създава един таб, отново с адрес "*about:blank*". И в двата случая този таб става **текущ**.

Ако потребителят **отвори нов таб**, той трябва да се добави непосредствено след текущия и на свой ред да стане текущ.

Ако **затворим даден таб**, текущ става този, който е непосредствено след него, а ако няма такъв – този преди него. При ситуацията, в която имаме само един отворен таб и го затворим, важи правилото описано по-нагоре в текста.

Вашата задача е да реализирате отварянето и затварянето на табове, посещаването на страници и прехвърлянето от един таб на друг.

Структури от данни
Домашна работа №1

За решаване на задачата трябва да се реализира програма с **команден интерфейс**, в която могат да се въвеждат следните команди:

GO <url>	Командата зарежда страницата с URL <url> в текущия таб. Автоматично трябва да се обнови timestamp-а на таба.
INSERT <url>	Добавя нов таб след текущия, в който се зарежда страницата с URL <url>. За този нов таб автоматично се обновява неговия timestamp. Новият таб става текущ.
BACK	Потребителят се прехвърля на таба, който е непосредствено преди текущия. Ако такъв няма (*т.е. в момента сме върху първия таб*), не се случва нищо.
FORWARD	Браузърът отива на таба, който е непосредствено след текущия. Ако такъв няма (*т.е. в момента сме върху последния таб*), не се случва нищо.
REMOVE	Премахва текущия таб. Текущ става този след него. Ако такъв няма, текущ става този преди него. Ако това е бил единственият отворен таб, се отваря нов и в него се зарежда адрес "*about:blank*".
PRINT	Извежда на екрана всички табове. Форматът е следния: <URL> <timestamp> (*Между двата елемента има един интервал*) Редът, на който се намира текущият таб, трябва да започва със знак за по-голямо (>), например: > www.example.com 123456789 Страниците да се извеждат точно в реда, в който се пазят в браузъра.

Освен тях може да се реализира и следната **опционална команда** (*тя носи допълнителни точки към решението*):

SORT <by>	Сортира всички табове лексикографски и прави текущ първия в наредбата. <by> може да приема две стойности: • **URL** – сортира табовете по URL, като ако има два с еднакъв URL, те се подреждат по timestamp; • **TIME** – сортира табовете по техния timestamp, като ако има два с еднакъв timestamp, те се подреждат по URL.

Пояснения:

Когато реализирате програмата, изберете подходящ **контейнер** от изучаваните в рамките на курса, в който да се пази информацията за табовете и върху който да се изпълняват операциите.

Ако реализирате и опционалната команда за сортиране, изберете подходящ **алгоритъм за сортиране**.

Изборът на структура от данни и начинът, по който ще работите с нея (*вкл. сортирането, ако решите да го реализирате*), трябва да бъдат съобразени с операциите, които ще извършва програмата ви. Помислете както за **времевата**, така и за **пространствената сложност**.

В задачата **НЕ** могат да се използват готови контейнери. Вместо това трябва сами да реализирате нужните алгоритми и структури от данни. Не е необходимо да реализирате клас за контейнер (*например клас за списък, масив, стек и т.н.*), освен ако не ви трябва такъв. Достатъчно е да реализирате тази функционалност, която е нужна за решението, като обаче работите коректно с паметта и спазвате принципите за добър дизайн.

По-долу е даден **пример** за работа на програмата. Редовете, започващи с **$**, са за въвеждане на команди:

```
$ GO www.youtube.com/watch?v=dQw4w9WgXcQ
$ INSERT www.google.bg/search?q=google+plz+help+me+solve+my+data+structures+task
```

$ INSERT
www.9gag.com/gag/aAV83g9/when-you-realise-today-is-the-homework-assignment-deadline
$ INSERT www.en.wikipedia.org/wiki/Data_structure
$ BACK
$ BACK
$ REMOVE
$ PRINT
www.youtube.com/watch?v=dQw4w9WgXcQ 150676023
>www.9gag.com/gag/aAV83g9/when-you-realise-today-is-the-homework-assignment-deadli
ne 150670593
www.en.wikipedia.org/wiki/Data_structure 150700011
$ REMOVE
$ GO www.wikihow.com/Deal-With-Tons-of-Homework
$ FORWARD
$ GO www.susi.uni-sofia.bg
$ PRINT
www.youtube.com/watch?v=dQw4w9WgXcQ 150676023
>www.susi.uni-sofia.bg 163750320

В случай, че реализирате опционалната команда за сортиране, тя може да работи например така:

$ GO www.youtube.com/watch?v=dQw4w9WgXcQ
$ INSERT www.en.wikipedia.org/wiki/Data_structure
$ INSERT www.wikihow.com/Deal-With-Tons-of-Homework
$ INSERT www.susi.uni-sofia.bg
$ SORT URL
$ PRINT
> www.en.wikipedia.org/wiki/Data_structure 150700011
www.susi.uni-sofia.bg 163750320
www.wikihow.com/Deal-With-Tons-of-Homework 150787241
www.youtube.com/watch?v=dQw4w9WgXcQ 150676023
$ SORT TIME
$ PRINT
>www.youtube.com/watch?v=dQw4w9WgXcQ 150676023
www.en.wikipedia.org/wiki/Data_structure 150700011
www.wikihow.com/Deal-With-Tons-of-Homework 150787241
www.susi.uni-sofia.bg 163750320

Структури от данни
Домашна работа №1

Задача 2.

Да се реализира специализация на шаблона на класа **Vector** за съхранение на булеви стойности (bool). Този клас трябва да представя контейнер от булеви стойности, който да работи с по-малко памет, като съхранява всяка стойност в един бит.

Изисквания към имплементацията:

1. **Методи за добавяне и премахване на елементи:**
   - **void push_back(value)** — добавя елемент в края на вектора.
   - **void pop_back()** — премахва последния елемент във вектора.
   - **void insert(iter, value)** — добавя елемент на позицията, посочена от итератора.
   - **void remove(iter)** — премахва елемент от позицията, посочена от итератора.
   - **void pop_front()** — премахва първия елемент във вектора.
2. **Индексация:**
   - Дефинирайте оператор [] за достъп/промяна на елементи по индекс. Той трябва да има константна и неконстантна версия.
3. **Итератори:**
   - Класът трябва да има итератори, които позволяват обхождане на всички елементи във вектора.
   - Итераторите трябва да поддържат основните операции: ++, --, *, !=, ==.
   - Трябва да имате **поне** следните итератори: **iterator**, **const iterator**, **reverse iterator**.
4. **Оптимизация на паметта:**
   - Класът трябва да съхранява елементите по оптимизиран начин, като всяка булева стойност заема само един бит.