13С112АСП2 – ДЗ3 Рок за израду: 09.01.2019.

ОДСЕК ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО

АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА 2

2018-2019

- трећи домаћи задатак -

Опште напомене:

1. Домаћи задатак 3 састоји се од два програмска проблема. Студенти проблеме решавају **самостално**, на програмском језику C++.
2. Реализовани програми треба да комуницирају са корисником путем једноставног менија који приказује реализоване операције и омогућава сукцесивну примену операција у произвољном редоследу.
3. Унос података треба омогућити било путем читања са стандардног улаза, било путем читања из датотеке.
4. Решења треба да буду отпорна на грешке и треба да кориснику пруже јасно обавештење у случају детекције грешке.
5. Приликом оцењивања, биће узето у обзир рационално коришћење ресурса. **Примена рекурзије се неће признати као успешно решење проблема.**
6. За све недовољно јасне захтеве у задатку, студенти треба да усвоје разумну претпоставку у вези реализације програма. Приликом одбране, демонстраторе треба обавестити која претпоставка је усвојена (или које претпоставке су усвојене) и која су ограничења програма (на пример, максимална димензија низа и слично). Неоправдано увођење ограничавајуће претпоставке повлачи негативне поене.
7. Одбрана трећег домаћег задатка ће се обавити у **среду, 09.01.2019. и четвртак, 10.01.2019.** према распореду који ће накнадно бити објављен на сајту предмета.
8. Пре одбране, сви студенти раде тест знања за рачунаром коришћењем система Moodle (http://elearning.rcub.bg.ac.rs/moodle/). **Сви студенти треба да се пријаве на курс пре почетка лабораторијских вежби**. Пријава на курс ће бити прихваћена и важећа само уколико је студент регистрован на систем путем свог налога електронске поште на серверу mail.student.etf.bg.ac.rs.
9. Формула за редни број **i** комбинације проблема коју треба користити приликом решавања задатка је следећа:

   (**R** – редни број индекса, **G** – последње две цифре године уписа):

   i = (R + G) mod 2
10. Имена датотека које се предају морају бити **dz3p1.cpp**
11. Предметни наставници задржавају право да изврше проверу сличности предатих домаћих задатака и коригују освојени број поена након одбране домаћих задатака.

26.12.2018. године *Са предмета*

Задатак 1 – Имплементација хипа [100 поена]

Написати на језику C++ потребне класе за реализацију хипа који садржи целобројне кључеве. Приликом стварања хипа, корисник бира да ли је то *max-heap* или *min-heap*. Операције које треба реализовати су:

- **[5 поена]** `void init()` - стварање новог (празног) хипа
- **[10 поена]** `void add(int elem, int &steps)` - додавање елемента у хип
- **[5 поена]** `int get()` - дохватање максималног, тј. минималног елемента хипа
- **[10 поена]** `int delete(int &steps)` - брисање максималног, тј. минималног елемента
- **[20 поена]** `union(Heap h, int &steps)` - додавање једног хипа другом (спајање хипова) – резултат је измењен хип за који се операција позива
- **[20 поена]** `convert()` - претварање *min*-хипа у *max*-хип, или обрнуто
- **[10 поена]** `void destroy()` - брисање (уништавање) хипа

Свако спајање, уметање и брисање треба да врати и колико је корака било потребно да би хип добио свој коначни изглед. У кораке се броје додавање стабла у листу коренова, поређење коренова стабала и спајање стабала истог реда.

У зависности од редног броја проблема који се решава, реализовати:

0. Фибоначијев хип
1. Биномни хип

Исправна реализација хипа подразумева да, поред наведених метода, постоје друге потребне методе (попут конструктора и деструктора). Студентима се препушта да у класу додају оне методе које сматрају потребним за успешну реализацију. Програм такође треба да води рачуна о исправном коришћењу меморије.

Главни програм и тестирање

[10 поена] Функција за тестирање користи реализован хип за имплементацију приоритетног реда. Приоритетни ред подржава функције:

- Стварања реда
- Испитивања да ли је ред празан
- Дохватање првог елемента реда
- Брисање (са или без дохватања) првог елемента реда
- Уметање елемента у ред

Приликом стварања реда, корисник уноси да ли мањи број представља мањи или већи приоритет.

Реализовати главни програм са једноставним интерактивним менијем који кориснику омогућава рад са јавним методама. Главни програм треба да омогући учитавање елемената из командне линије или датотеке, као и стварање случајних вредности за унос.

[10 поена] Главни програм треба да омогући и мерење перформанси. Перформансе треба дати у следећем табеларном облику:

Величина скупа података	Време уметања	Број корака приликом уметања	Време избацивања	Број корака приликом избацивања
10				
100				
1000				
10 000				
100 000				

Величина скупа података је број елемената који треба генерисати или учитати из датотеке. Време уметања представља време потребно за убацивање елемент по елемент из датог скупа у иницијално празан хип. Број корака приликом уметања је сума броја корака за сваки убачени елемент. Време избацивања представља време потребно да се из тог формираног хипа избацује елемент по елемент (минимални или максимални, у зависности од хипа). Број корака приликом избацивања је сума броја корака за сваки избачени елемент.

Прилог 1 - Рад са датотекама у језику C++

Рад са датотекама у језику C++ захтева увожење заглавља **`fstream`** (именски простор **`std`**). За читање података користи се класа **`ifstream`**. Након отварања датотеке, читање се врши на исти начин као и са стандардног улаза. Кратак преглед најбитнијих метода и пријатељских функција ове класе је дат у наставку.

`void open(`
`  const char *_Filename,`
`  ios_base::openmode _Mode = ios_base::in,`
`  int _Prot = (int)ios_base::_Openprot`
`);`
	Отвара датотеку задатог имена за читање.
	`ifstream dat;`
	`dat.open("datoteka.txt");`

`void close();`	Затвара датотеку.
`bool is_open();`	Утврђује да ли је датотека отворена.
`operator>>`	Преклопљен оператор за просте типове података.

`ifstream dat;`
`dat.open("datoteka.dat");`
`if( ! dat.is_open() )  greska();`
`char niz[20];`
`dat >> niz;`
`dat.close();`
	Пример отварања датотеке, провере да ли је отварање успешно, читање једног знаковног низа из датотеке и затварања датотеке.

Прилог 2 - Изворни код класе PerformanceCalculator

Класа је писана за оперативни систем Windows, под развојним окружењем Visual Studio 2008. Употреба класе се своди на 3 методе: **`start()`** да би се отпочело мерење, **`stop()`** да би се мерење прекинуло и **`elapsedMillis()`** да би се дохватио број милисекунди који је протекао између позива метода **`start()`** и **`stop()`**.

Пример примене:

```
#include "PerformanceCalculator.h"
void main(){
PerformanceCalculator pc;

	pc.start();
	Sleep(100);
	pc.stop();
	double protekloVreme = pc.elapsedMillis();
}
```

Изворни код класе *PerformanceCalculator* (датотека: PerformanceCalculator.h)

```
#ifndef PERFORMANCE_CALCULATOR_H__
#define PERFORMANCE_CALCULATOR_H__

#include "windows.h"

class PerformanceCalculator {
	LARGE_INTEGER	startCounter;
	LARGE_INTEGER	stopCounter;
	LARGE_INTEGER	frequency;
public:
	PerformanceCalculator() {
		startCounter.QuadPart = stopCounter.QuadPart = 0;
		QueryPerformanceFrequency(&frequency);
	}

	void start() {
		QueryPerformanceCounter(&startCounter);
	}

	void stop()  {
		QueryPerformanceCounter(&stopCounter);
	}

	double elapsedMillis() const {
		return (stopCounter.QuadPart-startCounter.QuadPart)*1000./frequency.QuadPart;
	}
};

#endif
```