ОДСЕК ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО
АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА 2

2025-2026

- први домаћи задатак -

Опште напомене:

1. Домаћи задатак 2 састоји се од једног програмска проблема. Студенти проблем решавају **самостално**, на програмском језику C++.
2. 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.
3. Реализовани програм треба да комуницира са корисником путем једноставног менија који приказује реализоване операције и омогућава сукцесивну примену операција у произвољном редоследу.
4. Унос података треба омогућити било путем читања са стандардног улаза, било путем читања из датотеке.
5. Решења треба да буду отпорна на грешке и треба да кориснику пруже јасно обавештење у случају детекције грешке.
6. Приликом оцењивања, биће узето у обзир рационално коришћење ресурса. **Примена рекурзије се неће признати као успешно решење проблема које може освојити максималан број поена.**
7. 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 матрице и слично). Неоправдано увођење ограничавајуће претпоставке повлачи негативне поене.
8. Одбрана другог домаћег задатка ће се обавити у **уторак, 30.12.2025.** према распореду који ће накнадно бити објављен на сајту предмета.
9. Име датотеке која се предаје мора бити **asp2dz2p1.cpp**. Погрешно назван и/или предат домаћи задатак се кажњава са одузетих 25 поена.
10. Предметни наставници задржавају право да изврше проверу сличности предатих домаћих задатака и коригују освојени број поена након одбране домаћих задатака, као и да пријаве теже случајеве повреде Правилника о дисциплинској одговорности студената Универзитета у Београду Дисциплинској комисији Факултета. Током израде решења није дозвољена употреба алата вештачке интелигенције заснованих на великим језичким моделима (*ChatGPT*, *Github Copilot* и сл.).

Задатак 1 – Имплементација распоређивача процеса коришћењем 2-3-4 стабала [100 поена]

На модерним оперативним системима, више програма може бити покренуто истовремено. Такви програми се називају процеси. У оквиру оперативног система може бити покренут велики број процеса. Процеси се извршавају на централној процесорској јединици (енг. *Central Processing Unit* – CPU), односно њеним језгрима (енг. *core*). С обзиром на то да је број јединица за извршавање процеса релативно мали у односу на број могућих процеса, процеси међусобно конкуришу за извршавање на CPU и само мали број њих се може истовремено извршавати.

Из наведених разлога, у оквиру оперативног система мора постојати посебан модул – распоређивач процеса (енг. *process scheduler*) који доноси одлуку који процес ће бити извршаван у ком тренутку. Активни процеси у оперативном систему се уобичајено смештају у ред за чекање, одакле их распоређивач процеса бира на неки начин и додељује им процесор за извршавање. Распоређивач процеса обично има могућност да заустави извршавање процеса након неког времена и поново га врати у ред за чекање.

У оквиру овог задатка, посматраће се поједностављена варијанта *Completely Fair Scheduler*-а присутног код оперативног система *Linux*. Код овог распоређивача процеса, ред за чекање се имплементира коришћењем црвено-црног стабла. У наставку текста је дат опис начина распоређивања процеса, као и операција које се том приликом спроводе.

Да би се обезбедила подједнака шанса за извршавање свим процесима, процесима се приликом покретања, поред имена процеса (стринг до 256 знакова), задаје време потребно за завршетак (енг. *time to complete*) и максимално време чекања (енг. *maximum waiting time*). Приликом убацивања новог процеса у ред за чекање (представљеног црвено-црним стаблом), процесу се придружују још два времена: тренутно време чекања (енг. *waiting time*) и тренутно време извршавања (енг. *execution time*) које се постављају на нула. Ред за чекање представљен црвено-црним стаблом је уређен по тренутном времену чекања.

Приликом извршавања, сваки процес добија инкремент времена (енг. *time slice*) који користи за извршавање. Процес се извршава или пун инкремент времена или само онолико времена колико је потребно да се достигне време потребно за завршетак рада процеса.

Поновно распоређивање процеса се врши након истека инкремента времена или завршетка рада неког од процеса, све док има активних процеса у реду за чекање. Алгоритам распоређивања је следећи:

1. За извршавање се бира најлевљи чвор у стаблу и избацује из стабла.
2. Увећавају се тренутно време чекања и тренутно време извршавања изабраног чвора. Износ увећања је једнак вредности инкремента времена за извршавање или износу преосталог времена потребног да изабрани процес заврши свој рад.
3. Врши се ажурирање свих чворова стабла увећавањем тренутног времена чекања. Уколико је тренутно време чекања неког процеса достигло или престигло максимално време чекања, од тренутног времена чекања се одузима максимално време чекања, чвор се избацује и поново умеће на одговарајуће место у стаблу.
4. Уколико укупно време извршавања није достигло време завршетка, изабрани процес за извршавање се враћа у стабло на одговарајуће место.

Због једноставности имплементације, **ред за чекање имплементирати 2-3-4 стаблом** на основу правила за изоморфизам 2-3-4 стабала и црвено-црних стабала.

Реализовати следеће операције над 2-3-4 стаблом:

1. [**10 поена**] Стварање празног стабла и уништавање стабла
2. **[20 поена]** Претраживање и уметање
   - Претраживање стабла по тренутном времену чекања и извршавања процеса
   - Уметање новог процеса у стабло
3. **[5 поена]** Учитавање из датотеке
   - Читање и формирање стабла на основу података из датотеке: један ред датотеке садржи име процеса (стринг до 256 знакова) и два цела броја - време потребно за завршетак (енг. *time to complete*) и максимално време чекања (енг. *maximum waiting time*)
4. **[5 поена]** Читање и манипулација појединачним процесима
5. **[20 поена]** Брисање процеса из стабла
6. **[10 поена]** Исписивање стабла у излазни ток (оператор <<)
   - Исписивање 2-3-4 стабла
   - Исписивање у облику бинарног црвено-црног стабла
7. **[30 поена]** Симулацију рада распоређивача процеса
   - Симулација корак-по-корак (за један инкремент времена)
   - Комплетна симулација уз испис стабла по корацима у датотеку

У задатку по потреби реализовати и додатне методе, где је то примерено.

Корисник са програмом интерагује путем једноставног менија. Програм треба да испише садржај менија, а затим да чека да корисник изабере (унесе путем тастатуре) редни број неке од понуђених ставки, након чега, пре извршења, од корисника очекује да по потреби унесе додатне параметре. Поступак се понавља све док корисник у менију не изабере опцију за прекид програма.

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.