ОДСЕК ЗА РАЧУНАРСКУ ТЕХНИКУ И ИНФОРМАТИКУ

АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА

2023-2024

- домаћи задатак -

Опште напомене:

1. Домаћи задатак састоји се од једног програмског проблема. Студенти проблем решавају **самостално**, на програмском језику C++. **Није дозвољено коришћење готових структура података из STL и сличних библиотека.**
2. Право да раде домаћи задатак имају сви студенти који прате предмет. Предаја домаћих задатака ће бити могућа до **понедељка, 11.12.2023.** коришћењем система *Moodle* (http://elearning.rcub.bg.ac.rs/moodle/). **Сви студенти треба да се пријаве на курс пре термина одбране домаћег задатка**. 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*. Прецизније информације везане за пријаву и предају домаћег ће бити накнадно објављене.
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 путем читања било са стандардног улаза, било из текстуалне 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 и неће бити оцењена са максималним бројем 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 низа и слично). Неоправдано увођење ограничавајуће претпоставке повлачи негативне поене.
8. Одбрана домаћег задатка ће се обавити у **понедељак, 11.12.2023.** према распореду који ће накнадно бити објављен на сајту предмета.
9. Формуле за редне бројеве проблема **i** и **j** које треба решавати су следеће (R – редни број индекса, G – последње две цифре године уписа):

$$\mathbf{i = (R + G) \bmod 2 + 1}$$

$$\mathbf{j = (R + G) \bmod 4 + 1}$$

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.

01.12.2023. године *Са предмета*

Стабла и графови [100 поена]

Програмски стек

У циљу боље анализе извршавања програма користе се бројни алати који прате и бележе токове извршавања програма. Један начин праћења извршавања програма је снимање садржаја програмског стека у различитим тренуцима извршавања програма. Програмски стек садржи функцију која се у том тренутку извршава, као и транзитивне позиваоце те функције (функцију која је позвала функцију која се извршава, затим и њеног позиваоца итд. до улазне програмске функције). Функције су на програмском стеку поређане тако да се улазна функција програма налази на дну стека, а позване функције се ређају ка врху стека тако да је на крају смештена функција чије се тело извршава у тренутку снимања.

На слици 1 дат је пример на основу кога ће бити илустрован појам програмског стека. Главна функција програма, од које почиње извршавање је функција *main*. Она позива функцију *a*, чије тело садржи само позив функције *b*. Функција *b* позива функцију *c*. Тренутак снимања садржаја програмског стека означен је звездицама (***) и дешава се у тренутку извршавања функције *c*. Садржај програмског стека је тада *main* > *a* > *b* > *c*.

```
void main(){
    ...
    a();
    ...
}

void a(){
    b();
}

void b(){
    c();
}

void c(){
    ***
    ...
}
```

Слика 1. Пример програма

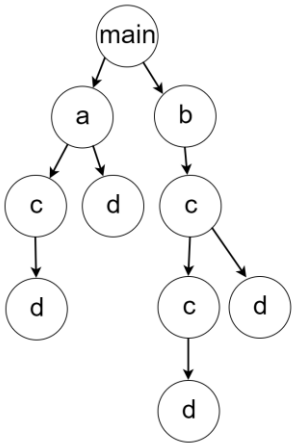
Формирање и манипулација стаблом [60 поена]

На основу скупа снимљених садржаја програмског стека могуће је формирати стабло извршавања програма. Сваки чвор стабла одговараће по једној функцији програма из засебне листе садржаја програмског стека, а релација родитеља и потомка се формира на основу релације позивалац - позвани. Корен датог стабла одговара главној функцији програма. Уколико су у стаблу претходно алоцирани, на одговарајућој позицији, чворови за део садржаја стека позива који се додаје, не треба их поново алоцирати.

Ради једноставности сматрати да је снимање садржаја стека извршено увек у оквиру функције чије тело не садржи позиве других функција.

На слици 2 приказан је сложенији пример програма. Као и у претходном, у овом примеру звездице означавају тренутак снимања садржаја програмског стека.

```
void main(){
    ...
    a();
    ...
    if (true){
        b();
    } else {
        c(2);
    }
}

void a(){
    d();
    c(0);
}

void b(){
    c(1);
}

void c(int num){
    if(num > 0){
        c(num-1);
    }
    d();
}

void d(){
    ***
    ...
}
```



Слика 2. Сложенији пример програма и одговарајуће стабло

Снимљени програмски стекови за пример са слике 2 су следећи:

main > a > c > d; main > a > d; main > b > c > c > d; main > b > c > d;

Резултујуће стабло у коме су смештени ови програмски стекови приказано је на десној страни исте слике.

Написати програм на програмском језику C++ који илуструје рад са стаблом. Програм треба да омогући следеће операције:

- **[15 поена]** Учитавање скупа програмских стекова из текстуалног фајла задатог формата и формирање стабла
- **[10 поена]** Додавање новог програмског стека у стабло
- **[10 поена]** Уклањање програмског стека из стабла
- **[15 поена]** Испис стабла
- **[10 поена]** Брисање стабла из меморије

Формат текстуалног фајла из којег се читају подаци је следећи. Садржај сваког појединачног програмског стека смештен је у засебном реду фајла. Имена функција су низови знакова који се састоје од бар једног малог слова и међусобно су одвојена размаком, а поређана су као у горњем примеру.

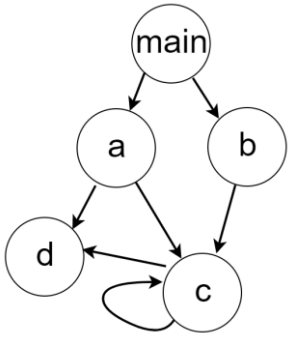
У зависности од редног броја **i** добијеног коришћењем формуле назначене у напоменама, потребно је користити један од следећих обилазака стабла приликом решавања задатих проблема:

1. *preorder order*
2. *level order*

Формирање и манипулација графом [40 поена]

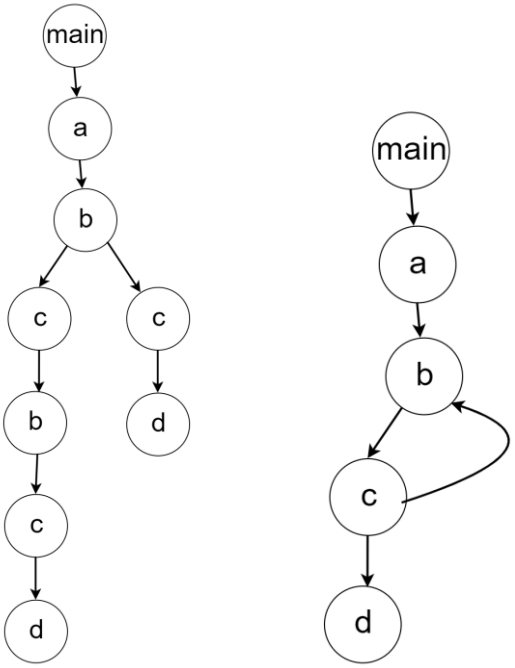
Ради уштеде простора стабло је могуће трансформисати у граф. На основу слике примера 2 могуће је формирати граф са слике 3. За разлику од стабла, у графу свакој функцији програма одговара тачно један чвор. Формирани граф ће садржати све снимљене стекове које садржи стабло, али и неке стекове које стабло не садржи (нпр. *main* > *a* > *c* > *c* > *d*).

Граф може садржати циклусе у случају рекурзивних позива. У примеру са слике 3 функција *c* позива себе, тако да граф садржи усмерену грану од чвора *c* до истог тог чвора, пошто је у питању директна рекурзија. У примеру са слике 4 постоји индиректна рекурзија (функције *b* и *c* се наизменично позивају), а резултујући граф садржи циклус који укључује чворове *b* и *c*.



Слика 3. Граф извршавања програма

```
void main(){
    a();
}

void a(){
    b(2);
}

void b(int num){
    if (num > 0){
        num--;
        c(num);
    }
}

void c(int num){
    d();
    b(num);
}

void d () {
    ***
}

main > a > b > c > d
main > a > b > c > b > c > d
```



Слика 4. Пример индиректне рекурзије

Потребно је имплементирати следеће функционалности:

- **[15 поена]** Конверзију формираног стабла у граф
- **[10 поена]** Испис графа на погодан начин
- **[15 поена]** Детекцију рекурзивних позива функција.

У зависности од редног броја **j** добијеног коришћењем формуле назначене у напоменама, потребно је користити једну од следећих меморијских репрезентација графа и један од два обиласка графа приликом решавања задатих проблема:

1. Матричну репрезентацију коришћењем матрица суседности и *BFS* алгоритам обиласка
2. Уланчану репрезентацију коришћењем листа суседности и *BFS* алгоритам обиласка
3. Уланчану репрезентацију коришћењем листа суседности и *DFS* алгоритам обиласка
4. Матричну репрезентацију коришћењем матрица суседности и *DFS* алгоритам обиласка

Више информација о наведеним меморијским репрезентацијама графа се може пронаћи у материјалима са предавања и вежби, као и у књизи проф. Мила Томашевића „Алгоритми и структуре података“.

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. Све наведене операције треба реализовати путем одговарајућих потпрограма чији је један од аргумената показивач на структуру података са којом се ради.