OBJEKTNO ORIJENTISANO PROGRAMIRANJE

- domaći zadatak broj 2 -

Funkcionalna specifikacija

Na programskom jeziku C++ implementirati mašinu za izvršavanje jednostavnih programa (*Machine*).

Učitavanje programa

Implementirati operaciju `void Machine::loadProgram(const string& filepath);` koja u mašinu učitava program čiji se opis nalazi u ulaznom fajlu. Putanja do fajla je zadata parametrom `filepath`. Ukoliko je prethodno već učitan program, sve informacije o starom programu se brišu i učitava se novi program. Format tekstualnog opisa programa je dat u nastavku.

- Program se sastoji iz proizvoljnog broja redova. Svaki red u programu sadrži opis jedne naredbe.
- Naredba se sastoji iz naziva naredbe i operanada koji su odvojeni sa po jednim znakom razmaka.
- Operand se sastoji iz naziva promenljive ili označene celobrojne vrednost.
- Naziv promenljive se sastoji iz jednog velikog slova.

Naredbe programa

U programu su podržane naredbe čiji su opisi dati u tabeli ispod. Predvideti i maksimalno pojednostaviti dodavanje novih vrsta naredbi (sa novim opisima, sa većim brojem operanada, itd.).

naziv naredbe	operand 1	operand 2	operand 3	opis	primeri
SET	promenljiva	promenljiva ili vrednost	/	Postavlja vrednost promenljive navedene kao prvi operand na vrednost drugog operanda.	`SET A -5` `SET B A`

ADD
SUB
MUL
DIV
	promenljiva	promenljiva ili vrednost	promenljiva ili vrednost	Sabira/Oduzima/Množi/Deli drugi i treći operand i smešta rezultat u promenljivu navedenu kao prvi operand. U slučaju deljenja, treći operand ne sme da bude 0.	`ADD A B 5`
					`SUB C 9 3`
					`MUL D -1 C`
					`DIV E D B`

GOTO	vrednost	/	/	Prelazi na sledeću naredbu koja se nalazi onoliko mesta od trenutne naredbe kolika je vrednost prvog operanda. Vrednost operanda mora da bude različita od nule.	`GOTO -1`
IFGR **IFEQ**	promenljiva ili vrednost	promenljiva ili vrednost	/	Izvršava blok naredbi između IFGR/IFEQ i ELSE naredbi ukoliko je prvi operand veći/jednak od drugog, a blok naredbi između ELSE i ENDIF naredbi, u suprotnom. Voditi računa o ugnežđenim uparivanjima.	`IFGR A 0` `IFEQ A B`
ELSE **ENDIF**	/	/	/	Ove naredbe su uparene sa nekom od prethodnih IFGR/IFEQ naredbi i nemaju nikakav efekat.	`ELSE` `ENDIF`
LOOP	/	/	/	Blok naredbi između LOOP i ENDLOOP naredbi se izvšrava beskonačno puta. Voditi računa o ugnežđenim uparivanjima.	`LOOP`
LOOP	vrednost	/	/	Blok naredbi između LOOP i ENDLOOP naredbi će se izvršiti onoliko puta kolika je vrednost prvog operanda. Voditi računa o ugnežđenim uparivanjima. Vrednost operanda mora da bude pozitivan broj.	`LOOP 5`
ENDLOOP	/	/	/	Ova naredba je uparena sa nekom od prethodnih LOOP naredbi i nema nikakav efekat.	`ENDLOOP`

Izvšravanje programa

Implementirati operaciju `void Machine::execute(const string& filepath);` koja izvršava učitani program i ispisuje vrednosti svih promenljivih u alfabetskom poretku u izlazni fajl. Svaka promenljiva i njena vrednost se pišu u zasebnom redu, odvojene znakom jednakosti. Putanja do izlaznog fajla je zadata parametrom `filepath`.

U nastavku je dat primer ulaznog fajla sa opisom programa koji računa najveći zajednički delilac broja A i B i smešta ga u promenljivu C. Izvršavanjem programa generiše se dati izlazni fajl.

Ulazni fajl	Izlazni fajl

SET A 18
SET B 24
LOOP
IFEQ B 0
GOTO 9
ELSE
ENDIF
IFGR A B
SUB A A B
ELSE
SUB B B A
ENDIF
ENDLOOP
SET C A
	A=6
	B=0
	C=6

Obrada grešaka

Neregularne situacije obrađivati konceptom izuzetaka. Prilikom obrade greške, ispisati grešku na standardni izlaz. Ispis treba što bolje da opisuje grešku (redni broj linije u fajlu, tip greške, itd.).

Test funkcija

Javni test sadrži funkciju `void test();` koja testira rad mašine. Studentima je javno dostupna implementacija funkcije `test` i mogu da je menjaju da bi dodatno testirali svoj kod kao što je opisano komentarima u kodu. Studentima su dati ulazni fajlovi koji predstavljaju test primere kao i izlazni fajlovi koji predstavljaju očekivane izlaze izvršavanja programa.

Tehnički zahtevi i smernice za izradu rešenja

Programski sistem realizovati tako da bude maksimalno efikasan, detaljno komentarisan, modularan i lako proširiv novim klasama i operacijama. Klasa *Machine* i njene operacije moraju biti imenovane prema zahtevima iz domaćeg zadatka. Programski kod klasa rasporediti u odgovarajuće **.h** i **.cpp** fajlove. Iz kolekcije standardne biblioteke dozvoljeno je koristiti sledeće tipove: `<string>`, `<vector>`, `<list>`, `<stack>`, `<queue>`. Ukoliko u zadatku nešto nije dovoljno jasno definisano, treba usvojiti razumnu pretpostavku i na temeljima te pretpostavke nastaviti izgrađivanje svog rešenja.

VAŽNE NAPOMENE

Za uspešno odbranjen domaći zadatak potrebno je na odbrani pokazati kod podeljen na odgovarajuće projekte, .h i .cpp fajlove.

1. Klase kojima su implementirani osnovni koncepti treba da budu smeštene u poseban projekat rešenja koji se prevodi kao statička biblioteka (**`Machine.lib`**).
2. Glavni program treba da se nalazi u posebnom projektu koji se prevodi kao Win32 Console Application (**`testdz2.exe`**) fajl i koji treba povezati sa statičkom bibliotekom. Glavni program napraviti u fajlu **`Main.cpp`**. U glavni projekat je dodatno potrebno uključiti fajl **`Test.cpp`**.