Zadatak: “program”

Cilj nam je moći definirati “programe” koji mijenjaju brojeve u memoriji jednostavnog računala. Postojat će dvije moguće naredbe (s argumentima) u takvim programima: naredba koja mijenja vrijednost broja u označenoj memorijskoj ćeliji, i naredba kojom je moguće označiti drugu ćeliju.

U ovom zadatku, `memorija` je neprazan konačan slijed brojeva od kojih je u svakom trenutku točno jedan broj “označen” (na početku je odabran prvi broj).

Primjerice, ako je vektor `brojevi` neki vektor cijelih brojeva $\{1, 10, 100\}$, tada sljedeći dio programa ispisuje `1 12 107`.

```
memorija mem(brojevi);
program prog;

prog << new pomak(2)
     << new inkrement(7)
     << new pomak(-1)
     << new inkrement(2);

int celija = prog(mem);

cout << mem[0] << '␣' << mem[1] << '␣' << mem[2] << endl;
```

Opširniji testni klijentski program nalazi se na dnu zadatka.

Potrebno je implementirati strukture `memorija`, `naredba`, `pomak` (podtip od `naredba`), `inkrement` (podtip od `naredba`) i `program`.

Struktura `memorija` pamti kolekciju cijelih brojeva (`int`) te informaciju o trenutno odabranoj ćeliji memorije. Postoji konstruktor koji prima `vector<int>`, i tako primljenom kolekcijom elemenata inicijalizira memoriju. Osim toga, `memorija` definira i operator indeksiranja (`operator[]`) koji prima indeks te vraća referencu na broj spremljen u kolekciji na tom indeksu.

Struktura `naredba` pamti jedan cijeli broj (`int`). Taj broj zovemo argumentom naredbe. Struktura `naredba` definira čisti virtualni `operator()` koji uzima jedan parametar: referencu na objekt tipa `memorija`, i ništa ne vraća. Definira i operator pretvorbe u `int`. Taj operator vraća argument naredbe.

Struktura `pomak` je podtip strukture `naredba`. Nadograđuje virtualni `operator()` tako da on pomiče odabir memorijske ćelije za iznos argumenta naredbe (varijabla članica strukture `naredba`). Mora postojati konstruktor koji uzima cijeli broj (`int`) i njime inicijalizira argument naredbe.

Struktura `inkrement` je također podtip strukture `naredba`. Ova struktura nadograđuje virtualni `operator()` tako da on vrijednost odabrane memorijske ćelije mijenja (relativno) za iznos argumenta naredbe (varijabla članica strukture `naredba`). Mora postojati konstruktor koji uzima cijeli broj (`int`) i njime inicijalizira argument naredbe. Osim toga, `inkrement` definira i operator pridruživanja koji primljeni niz znakova (`const char*`) sastavljen od nepraznog niza znamenaka (iz skupa $\{0, \ldots, 7\}$) interpretira kao zapis nekog (nenegativnog cijelog) broja u sustavu s bazom 8 te tako pročitan broj sprema kao argument naredbe. Taj operator vraća referencu na objekt koji je odredište pridruživanja.

Struktura `program` mora moći pamtiti pokazivače na naredbe. Definira i `operator()` koji prima referencu na objekt tipa `memorija` i slijedno izvršava sve naredbe programa. Pozi-

vatelju vraća finalnu vrijednost prve memorijske ćelije. Osim toga, definira i `operator<<` kojim se dodaje nova naredba. Taj operator prima objekt tipa `naredba*`.

Sučelje struktura koje definirate spremite u datoteku `program.h`. Implementaciju spremite također u `program.h`, ili u novu datoteku `program.cpp`. Implementacijsku datoteku predajte čak i ako je prazna, tj. ako ste implementaciju napisali u datoteci `program.h`.

Testni primjer:

```
#include <iostream>
#include <vector>
#include "program.h"

using namespace std;

int main() {
  vector<int> v = {1, 10, 50}; /* ekvivalentno: v.push_back(1); v.push_back(10); v.push_back(50); */
  memorija mem(v);
  cout << mem[0] << '␣' << mem[1] << '␣' << mem[2] << endl; /* izlaz: 1 10 50 */

  mem[1] = mem[2] = 100;
  cout << mem[0] << '␣' << mem[1] << '␣' << mem[2] << endl; /* izlaz: 1 100 100 */

  program pr;

  pr << new pomak(2) << new inkrement(3)
     << new inkrement(0) << new inkrement(4)
     << new pomak(-1) << new inkrement(-16)
     << new pomak(-1) << new inkrement(0);

  inkrement jos_jedan_inkrement(0);
  jos_jedan_inkrement = "24";

  pr << new pomak(1) << &jos_jedan_inkrement;

  int prvi = pr(mem);
  cout << int(jos_jedan_inkrement) << '␣' << prvi << '␣'
       << mem[0] << '␣' << mem[1] << '␣' << mem[2] << endl; /* izlaz: 20 1 1 104 107 */

  return 0;
}
```

Napomene:

- Na pojedinom objektu tipa `program` najviše će se jednom pozvati operator pozivanja (`operator()`). Pojedini objekt tipa `memorija` bit će argument u najviše jednom takvom pozivu.
- Nigdje ne morate provjeravati smislenost parametara koji dolaze iz klijentskog dijela programa, npr. ne morate provjeravati jesu li indeksi u granicama.
- Pozivatelj nikad neće koristiti konstantne varijable bilo kojeg među spomenutim tipovima.
- Sve operatore možete definirati unutar definicija odgovarajućih struktura (ne morate koristiti globalne funkcije).
- U svojem rješenju nigdje ne morate dealocirati memoriju.

Ako imate dodatnih pitanja, javite na lmikec@math.hr