G.4. ConsoleConsole

Alăturat aveți codul sursă al unui program C++ proiectat conform arhitecturii (pattern) Model View Controller (MVC).

MVC Pattern is used to separate application's concerns.

- **Model** - Model represents an object carrying data. It can also have logic to update data changes.
- **View** - View represents the visualization of the data that model contains.
- **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

În acest exemplu:

- comenzile (add) de executare a operațiilor pe obiectul de tip Model sunt citite din fișierul input.txt. Acest fișier conține două linii, pe fiecare este înscrisă comanda add:

add

add

- rezultatul este înscris în fișierul output.txt.

a) Explicați rezultatele executării acestui program.

b) Se cere să scrieți o altă interfață View, astfel încât:
- comenzile să fie citite de la consola (tastatura, cin): vor fi scrise două linii cu textul add pe fiecare
iar
-rezultatele să fie afișate pe consolă (pe monitor, cout).

```
#include <iostream>
#include <fstream>
using namespace std;
class Model{
   public:
      Model():x(0){}
      int incr(){x++; return x;}
   private:
      int x;
};
class AbstractView{
public:
   virtual string get_command()=0;
   virtual void write_result(int i)=0;
};
```

```
class View: public AbstractView{


public:
   View(){
      inp.open("input.txt", ios::in);
      out.open ("output.txt",ios::out);


   }
   string get_command(){
      string rez;
      getline(inp,rez);
      return rez;
   }
   void write_result(int i){
      out<<i<<endl;
   }
   private:
      ifstream inp;
      ofstream out;


};
```

```
class Controller{
public:
   Controller(Model m, AbstractView *v){
       md=m;
       vw=v;
    }
    void process(){
      string comm=vw->get_command();
      if (comm=="add") {
         int r= md.incr();
         vw->write_result(r);
      }else vw->write_result(-1);
    }

private:
   Model md;
   AbstractView *vw;
};
```

```
int main()
{
   cout << "Model View Controller in C++" << endl;
   Model mod;
   View view;
   Controller cntrl(mod, &view);
   cntrl.process();
   cntrl.process();
   return 0;
}
```