Supraîncărcare operatori de I/O: operator<< , operator>>

- sens de baza: deplasare pe biti (aritmetic pe *signed*, logic pe *unsigned*)
- sens la supraîncărcare: **depunere / extragere** informații in/din fluxuri de I/O
- direcția săgeților indică sensul curgerii informației:
  - cout **<<** i; // din variabila i catre stream iesire
  - cin **>>** i; // din stream intrare catre variabila i
- ideea de continuitate flux prin compunere operator cu el însuși, prin transfer referință:

  cout << s1; // operator<<(cout, s1); // funcție independentă!

  cout << s1 << s2; // operator<<(**operator<<(cout, s1)**, s2);
- citire șir de caractere **cu spatii** (**char nume[30]**) :

  cin >> s.varsta; cin.ignore(); //eliminare <CR> de la citiri anterioare

  cout<<"\nNume: "; cin.getline(s.nume, 29);

  // citire sir de **max 29** caractere sau terminare cu <CR>
- Problema pe tip string: citire **string** nume; continand si spatii ?
- I/O sunt **asincrone**;
- sincronizare explicită cu **restul programului**:
  - *cin.**ignore()**; // fortare terminare citire prin eliminare caractere din flux*
  - *cout.**flush();** // programul asteapta pana se termina afisarea*
- sincronizarea celor doua mecanisme ***stdio*** si ***iostream***
  - *cout.**sync_with_stdio();***

Supraîncărcare operatori de I/O: IO pe fisiere nestandard

#include<fstream>

- Fisiere pe disc:
  - **text**: delimitare cu **endl** după nume (când conține spatii) pentru a sti unde se termină;
  - **binar**: **read / write** au nevoie de *sizeof(obiect)* evaluat la compilare;
    - poate fi calculat corect pentru char nume[30];
    - pentru char* si string la compilare nu stim ce vom stoca drept nume!
    - lungimea se calculeaza cu functii, la momentul executiei:
      - **strlen_s**(nume) la char*
      - **size() ; resize()** la string
    - lungimea trebuie scrisa in fisier, fiind necesara la citirea din fisier a unui camp de lungime variabila
- .

Supraîncărcare operatori de I/O: manipulare flux

```
#include<iomanip>
```

- fluxurile pot fi configurate altfel decât cum au valori implicite
- manipularea se poate face cu functii ce includ manipulatori:

```
ostream & moneyFormat(ostream & ies)
{
      ies << setw(15); ies << setfill('$');
      ies << setprecision(5);
      ies << setiosflags(ios::showpoint);
      ies << setiosflags(ios::fixed);
      return ies;
}
```

- ulterior apelul poate fi intercalat in flux:

```
cout << endl << moneyFormat << d;
```

- .