



UNIVERSITATEA BABEŞ-BOLYAI  
Facultatea de Matematică și Informatică



# Programare orientată obiect

---

Curs 12

Laura Dioşan

# POO

---

- Operații de intrare/ieșire (IO)
  - Fișiere de I/O
  - Biblioteci standard
    - Biblioteca Standard de I/O
    - Biblioteca Standard pentru *stream*-uri I/O

# Intrări și ieșiri

---

- ❑ Facilitățile I/O nu fac parte din limbajul C propriu-zis
- ❑ Toate operațiile de I/o trebuie realizate prin intermediul unor funcții speciale, precum
  - *getchar()*,
  - *scanf()*,
  - *printf()*, etc.
- ❑ Ele nu fac parte din limbajul propriu-zis → pot să difere de la o mașină la alta
- ❑ Standardul ANSI

# Fișiere IO

---

## □ Tipologie

### ■ Fișiere text

- Informația este reținută ca vectori de caractere
- Sfârșitul de fișier → există fizic în fișier (***fgetc()*** returns EOF (-1))

### ■ Fișiere binare

- Informația este reținută în formatul original (intern):
  - *int* → 2 octeți
  - *float* → 4 octeți, etc.
- Sfârșitul de fișier → nu există fizic în fișier (***fgetc()*** îl va genera)

## □ Procesarea fișierelor la 2 nivele:

### ■ Nivel inferior

- Acces direct la SO

### ■ Nivel superior

- Utilizarea de funcții de I/O speciale

# Procesarea la nivel inferior

---

- Deschiderea unui fișier
- Închiderea fișierului
- Citirea din fișier
- Scrierea în fișier
- Accesul în fișier

# Procesarea la nivel inferior

---

## □ Deschiderea unui fișier

```
int open(const char* path, int access)
```

- acces
  - O\_RDONLY
  - O\_WRONLY
  - O\_CREATE
  - O\_RDWR
  - O\_APPEND
  - O\_BINARY
  - O\_TEXT
- return
  - descriptor de fișier
  - -1 – error
- headers:
  - <io.h>, <fcntl.h>

# Procesarea la nivel inferior

---

## □ Închiderea unui fișier

```
int close(int handle)
```

- handle
  - Descriptor de fișier
- return
  - 0 – închidere cu succes
  - -1 – eroare

# Procesarea la nivel inferior

---

## □ Citirea dintr-un fișier

```
int read(int handle, void* buffer, unsigned length)
```

- handle
  - 0 - stdin
  - 1 - stdout
  - 2 - stderr
  - 3 - stdprn
  - 4 - stdaux
- buffer
  - Zona de memorie în care se citește
- length
  - Numărul maxim de octeți citați
- return
  - # de octeți citați
  - 0 - la sfârșit de fișier
  - -1 - eroare



# Procesarea la nivel inferior

---

## □ Scrierea în fișier

```
int write(int handle, void* buffer, unsigned length)
```

- handle
  - 0 - stdin
  - 1 - stdout
  - 2 - stderr
  - 3 - stdprn
  - 4 - stdaux
- buffer
  - Zona de memorie din care se preia informația care trebuie afișată
- length
  - # maxim de octeți scriși
- return
  - # de octeți scriși
  - -1 - eroare

# Procesarea la nivel inferior

---

## □ Accesul în fișier

```
long lseek(int handle, long offset, int origin)
```

- handle
  - Descriptor de fișier
- offset
  - Numărul (cu semn) de octeți după care se face poziționarea în fișier (relativ la origine)
- origin
  - Poziția referință
    - SEEK\_SET → începutul fișierului
    - SEEK\_CUR → poziția curentă
    - SEEK\_END → sfârșitul fișierului
- return
  - Numărul de octeți după care se face poziționarea în fișier (relativ la poziția originală)
  - -1 – eroare

# Procesarea la nivel superior

---

- ❑ Deschiderea unui fișier
- ❑ Închiderea fișierului
- ❑ Procesarea bazată pe caractere
- ❑ Procesarea bazată pe șiruri de caractere
- ❑ Procesarea bazată pe fluxuri (*stream*-uri)
- ❑ Citirea din fișier
- ❑ Scrierea în fișier
  
- ❑ ➔ Biblioteca Standard ➔ stil C
- ❑ ➔ Biblioteca pentru fluxuri de I/O - Standard Input / Output Streams Library ➔ stil C++

# Biblioteca standard

---

- ❑ O bibliotecă NE-orientată obiect care oferă funcționalități pentru operațiile de I/O
- ❑ Operații de I/O standard (consolă, respectiv tastatură)
- ❑ Operații de I/O cu fișiere

# Operații standard de I/O

---

## □ Operații de intrare

- *int* **getchar()**
  - Preia un singur caracter de la consolă
  - Întoarce caracterul de intrare sau EOF
- *int* **getc(FILE\*)**
  - Preia un singur caracter de la consolă sau dintr-un fișier
  - Întoarce caracterul de intrare sau EOF
- *char\** **gets(char\*)**
  - Citește un string de la consolă
  - Returnează un pointer către stringul citit sau NULL (în caz de eroare)

## □ Operații de ieșire

- *int* **putchar(int)**
  - Scrie un caracter pe consolă
  - Returnează caracterul scris în caz de succes sau EOF în caz de eroare
- *int* **putc(int, FILE\*)**
  - Scrie un caracter pe consolă sau într-un fișier
  - Returnează caracterul scris în caz de succes sau EOF în caz de eroare
- *int* **puts(char\*)**
  - Scrie un string pe consolă
  - Returnează un număr ne-negativ în caz de succes sau EOF în caz de eroare

```
char c;  
while ((c = getchar()) != EOF){  
    putchar(c);  
}
```

```
char c;  
while ((c = getc(stdin)) != EOF){  
    putc(c, stdout);  
}
```

```
char s[20];  
while (gets(s) != NULL)  
    puts(s);
```

# Operații standard de I/O

---

- Operații de intrare formatate
  - *int* **scanf**(*char\** format, *arg1*, *arg2*, ...)
    - Citește un string formatat de la consolă
    - Returnează numărul de elemente citite și care s-au potrivit cu formatul sau EOF
  - *int* **sscanf**(*char\** string, *char\** format, *arg1*, *arg2*, ...)
    - Citește un string formatat dintr-un alt string
- Formatted output operations
  - *int* **printf**(*char\** format, *arg1*, *arg2*, ...)
    - Tipărește un string formatat pe consolă
    - Returnează numărul de caractere tipărite
  - *int* **sprintf**(*char\** string, *char\** format, *arg1*, *arg2*, ...)
    - Tipărește un string formatat în alt string
- Format:
  - %d → valori întregi în baza 10
  - %o → valori întregi în baza 8
  - %x → valori întregi în baza 16
  - %X → valori întregi în baza 16 (upper-case)
  - %f → valori reale
  - %c → caractere
  - %s → stringuri

# Exemplu

---

- A se consulta exemplul din directorul *11/streams*, funcțiile:
  - *standardIO()*
  - *getPut()*

# Operații de I/O cu fișiere

---

- Deschiderea unui fișier
- Citirea din fișier
- Scrierea în fișiere
- Închiderea unui fișier



# Deschiderea unui fișier

---

- Declararea unei variabile de tip fișier
  - *FILE\* fileVariables*
- Deschiderea unui fișiere
  - *FILE \***fopen**(char\* fileName, char\* access);*
- Access:
  - "r" – deschidere pentru citire
  - "w" – deschidere pentru scriere
  - "a" – deschidere pentru adăugare
  - "r+" – deschidere pentru citire/scriere
  - "rb" – deschidere pentru citire în format binar
  - "wb" – deschidere pentru scriere în format binar
  - "r+b" – deschidere pentru citire/scriere în format binar

# Închiderea unui fișiere

---

- `int fclose(FILE *pf);`
  - returnează 0 – succes
  - returnează 1 – erori

# Operații de I/O cu fișiere

---

<b>Tip fișier</b>	<b>Conversie</b>	<b>Unitatea transferată</b>	<b>Funcții</b>
text	Fără	Character	<code>fgetc()</code> ⇔ <code>getc()</code> <code>fputc()</code> ⇔ <code>putc()</code>
		Linie	<code>fgets()</code> ⇔ <code>gets()</code> <code>fputs()</code> ⇔ <code>puts()</code>
	Cu	Linii	<code>fscanf()</code> <code>fprintf()</code>
binar	fără	Articol	<code>fread()</code> <code>fwrite()</code>

# Operații de I/O cu fișiere

---

- *int* **fscanf**(*FILE\** stream, *char\** format, *arg1* , *arg2*, ...)
  - Citește dintr-un fișier un string, caracter sau o dată numerică formatat(ă)
  - Returnează numărul de elemente corect citite
- *int* **fprintf**(*FILE\** stream, *char\** format, *arg1*, *arg2*, ...)
  - Afisează un string formatat într-un fișier
  - Returnează numărul de elemente afișate sau 0 (eroare)
- *size\_t* **fread**(*void \*p*, *size\_t elemSize*, *size\_t noOfElem*, *FILE \*stream*)
  - Citește dintr-un fișier binar
  - Returnează numărul de elemente corect citite
- *size\_t* **fwrite**(*void\* p*, *size\_t elemSize*, *size\_t noOfElem*, *FILE\* stream*)
  - Scrie într-un fișier binar
  - Returnează numărul de elemente scrise sau 0 (eroare)

# Exemple

---

- A se consulta exemplul din directorul *12/streams*, funcția:
  - *standardFileIO()*;

# Biblioteca standard pentru fluxuri de I/O (Standard IO stream library)

---

- ❑ O bibliotecă orientată obiect care oferă funcționalități pentru *stream*-urile de I/O
- ❑ *Stream*
  - O abstractizare care reprezintă un dispozitiv cu ajutorul căruia se efectuează operații de I/O
  - Poate fi reprezentat ca o sursă sau destinație formată din-un număr neprecizat de caractere
- ❑ Fișiere componente ale bibliotecii:
  - `<ios>`, `<istream>`, `<ostream>`, `<streambuf>` și `<iosfwd>`
    - ❑ clasele de bază
  - `<iostream>`
    - ❑ Declară obiecte folosite în comunicarea dintre intrarea și ieșirea standard
    - ❑ include obiecte precum *cin* și *cout*
  - `<fstream>`
    - ❑ Definiște clasele pentru lucrul cu fișiere
  - `<sstream>`
    - ❑ Pentru manipularea stringurilor din *stream*-uri
  - `<iomanip>`
    - ❑ Modifică *flag*-urile interne și opțiunile de formatare

# Elementele bibliotecii IOS

---

- Clase
  - Clase template de bază
  - Instanțieri ale claselor template
    - *Narrow-oriented*
    - *Wide-oriented*
- Obiecte standard
- Tipuri
- Manipulatori

# De ce *streams*?

---

- Citirea și afișarea în C
  - *scanf*
  - *printf*
  
  - Nu pot fi extinse
  - Tratează doar tipurile de date de bază (int, char, float, char\*, etc)
  
- → funcții pentru *stream*-uri



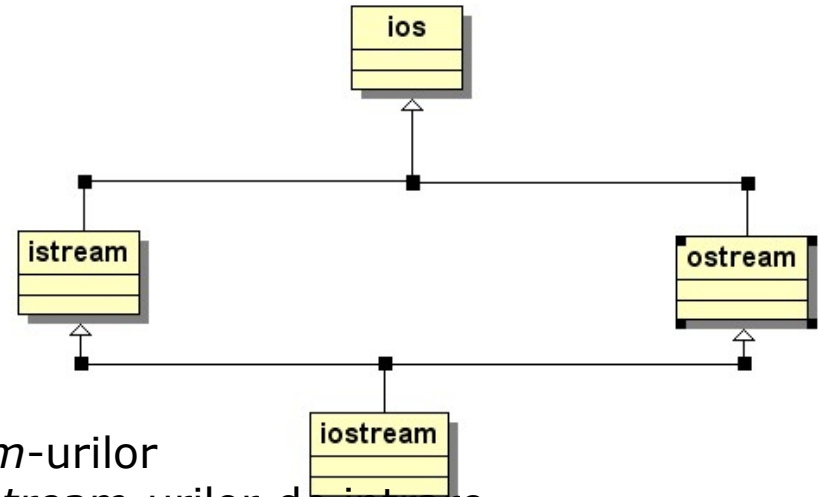
# Stream-uri

---

- *Stream* = un obiect care transportă și formatează caractere:
  - de la o sursă:
    - tastatură(stdin),
    - fișier sau
    - zonă de memorie
  - la o destinație:
    - ecran (stdout),
    - fișier sau
    - zonă de memorie.
  - Caracterele sunt procesate în mod serial (unele după altele)

# Ierarhia de clase

---



- *ios* → comportamentul general al *stream*-urilor
- *istream* → comportamentul general al *stream*-urilor de intrare
- *ostream* → comportamentul general al *stream*-urilor de ieșire
- *iostream* = *istream* + *ostream*
  
- Obiecte:
  - *cin* – un obiect de tip *istream* (*stream*-ul standard de intrare)
  - *cout* – un obiect de tip *ostream* (*stream*-ul standard de ieșire)
  - *cerr* – un obiect de tip *ostream* (*stream*-ul standard de eroare)
  - *clog* – un obiect de tip *ostream* (*stream*-ul standard de *logging*)
  
- Operatori
  - << (inserter)
  - >> (extracter)
    - Exclue spațiile atunci când *flag*-ul *ios::skipws* este setat (implicit)

# Starea unui stream

---

Setarea *flag*-ului unui *stream*: *streamObj.setstate(flag)*

```
void setFlag(){  
    ofstream os;  
    os.setstate(ios::failbit);  
}
```

<b>Flag</b>	<b>Explicație</b>
<i>fail</i>	Date invalide
<i>badbit</i>	Eroare fizică
<i>goodbit</i>	OK
<i>eofbit</i>	A fost detectat caracterul sfârșit de fișier (EOF)

# Starea unui *stream*

---

- Verificarea stării unui *stream*

<i>bool fail()</i>
<i>bool good()</i>
<i>bool eof()</i>
<i>bool bad()</i>

```
void checkTheFlag(){
    ofstream os;
    if (os.good()){
        //use os
    }
}
```

# Starea unui *stream*

---

- Resetarea tuturor *flag*-urilor: *streamObj.clear()*
  - Toate *flag*-urile sunt resetate
  - Funcția *good()* va returna *true*

```
void resetFlags(){
    ofstream os;
    os.setstate(ios::failbit);
    os.clear();
    bool b = os.good();           //b = true
}
```

# *Stream-uri - exemple*

---

- A se consulta exemplul din directorul *12/streams*, funcțiile:
  - *setFlag();*
  - *checkTheFlag();*
  - *resetFlags();*
  - *dataProcessing();*

# Intrări orientate pe linii

---

## □ Problemă

- `cin >>` - citirea se oprește la primul spațiu

## □ Soluția

- `istreamObj.get()`
- `istreamObj.getline()`
- `istreamObj.ignore()`

# istreamObj.get()

---

- ❑ Similar lui >>, dar cu 2 excepții:
  - Include și spațiile
  - e mai puțin probabil ca *stream*-ul de ieșire să fie curățat
  
- ❑ 3 versiuni:
  - ***istreamObj.get()***
    - ❑ citește primul caracter citit din *stream* și în returnează
    - ❑ celelalte caractere rămân în *stream*
  
  - ***istreamObj.get(bufferAdress, maxNoOfElem)***
    - ❑ citește din *stream* în *buffer* maximum *maxNoOfElem* caractere
    - ❑ celelalte caractere rămân în *stream*
  
  - ***istreamObj.get(bufferAdress, maxNoOfElem, delim)***
    - ❑ citește din *stream* în *buffer* maximum *maxNoOfElem* caractere până la întâlnirea primei apariții a delimitatorului *delim*
    - ❑ celelalte caractere rămân în *stream*
    - ❑ caracterul de terminare rămâne în *stream*



# istreamObj.getline(...)

---

- ❑ Similar cu *get*, dar cu 1 excepție:
  - Elimină caracterul de terminare din stream (pentru ultima versiune)
  
- ❑ 2 versiuni:
  - ***istreamObj.get(bufferAdress, maxNoOfElem)***
    - ❑ citește din *stream* în *buffer* maximum *maxNoOfElem* caractere
    - ❑ celelalte caractere rămân în *stream*
  
  - ***istreamObj.get(bufferAdress, maxNoOfElem, delim)***
    - ❑ citește din *stream* în *buffer* maximum *maxNoOfElem* caractere până la întâlnirea primei apariții a delimitatorului *delim*
    - ❑ celelalte caractere rămân în *stream*
    - ❑ elimină din *stream* caracterul de terminare

# istreamObj.ignore(...)

---

## □ ***istreamObj.ignore(count, delim)***

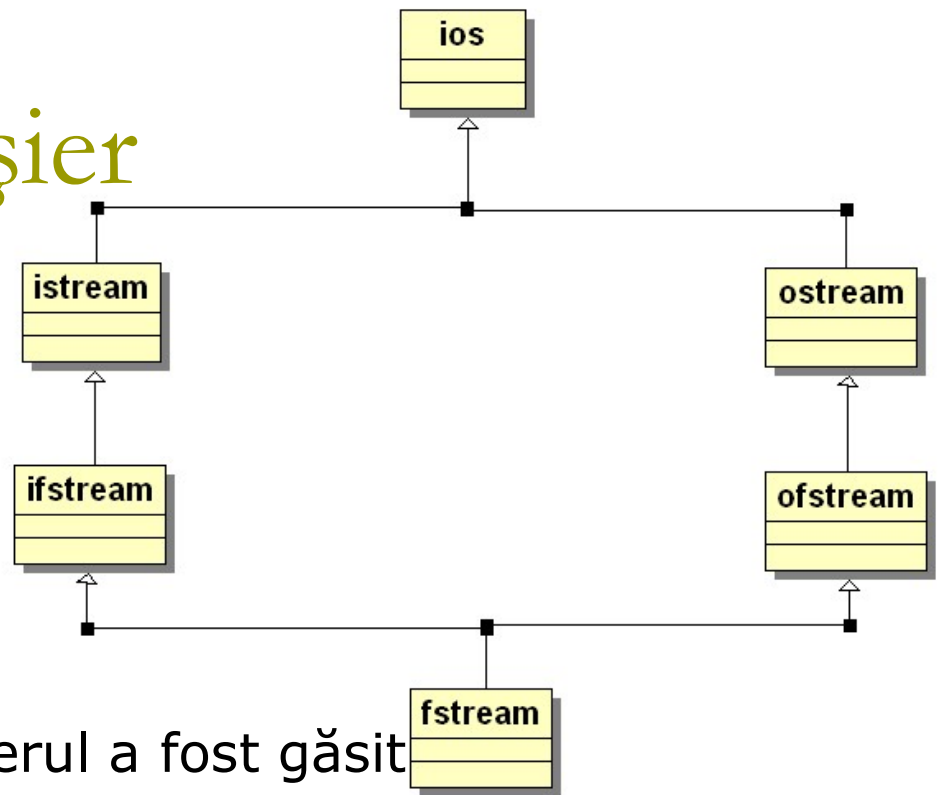
- extrage și descarcă din *stream* până la *count* caractere
- extragerea se oprește dacă
  - delimitatorul *delim* este extras
  - sfârșitul de *stream* este întâlnit
- caracterul de delimitare este extras din *stream*

# Exemplu

---

- A se consulta exemplul din directorul *12/streams*, funcțiile:
  - *getExample();*
  - *getlineExample();*

# Stream-uri de tip fișier



- *ifstream*
  - Trebuie verificat dacă fișierul a fost găsit
- *ofstream*
  - Fișierul trebuie închis
- A se consulta exemplul din directorul *11/streams*, funcția
  - *ifstreamExample()*;

# Deschiderea unui *stream* de tip fișier

- Cum?
  - Utilizarea constructorului
  - Utilizarea funcției *open(...)*
  - În ambele cazuri, există 2 posibilități:
    - Folosirea numelui fișierului
    - Folosirea numelui fișierului și a modului de deschidere (*flag*)

Flag	Explicație
<i>ios::in</i>	Deschidere pentru citire (valoare implicită pentru <i>ifstream</i> )
<i>ios::out</i>	Deschidere pentru scriere (valoare implicită pentru <i>ofstream</i> )
<i>ios::ate</i>	Setarea indicatorului de poziție la sfârșitul <i>stream</i> -ului în momentul deschiderii ( <i>at end</i> )
<i>ios::app</i>	Setarea indicatorului de poziție la sfârșitul <i>stream</i> -ului înainte de orice operație de scriere ( <i>append</i> )
<i>ios::binary</i>	Considerarea binară a <i>stream</i> -ului (în loc de text)
<i>ios::trunc</i>	Curățarea <i>stream</i> -ului → la deschidere, <i>stream</i> -ul va avea lungimea 0

# Închiderea unui *stream* de tip fișier

---

- Cum?
  - Folosirea (automată) a destructorului
  - Folosirea funcției *close()*

# Stream-uri tampon (*stream buffering*)

---

- `rdbuf(...)`
  - Get/set stream-ul tampon (*stream buffer*) asociat
  - `streambuf* rdbuf ( )`
    - Returnează obiectul de tip *stream buffer* asociat *stream-ului*
  - `streambuf* rdbuf ( streambuf* sb )`
    - Asociază stream-ul cu *sb* și returnează obiectul de tip *stream buffer* asociat anterior *stream-ului*
  
- A se consulta exemplul din directorul *12/streams*, funcția
  - `streamBuffering()`

# Poziționarea într-un fișier

---

## □ Poziția absolută

### ■ *ostream*:

□ ***tellp()*** → returnează poziția absolută

□ ***seekp()*** → setează poziția absolută

### ■ *istream*:

□ ***tellg()*** → returnează poziția absolută

□ ***seekg()*** → setează poziția absolută

## □ Poziția relativă

Flag	Explicație
<i>ios::beg</i>	Poziționare la început de fișier
<i>ios::cur</i>	Poziționare la poziția curentă din fișier
<i>ios::end</i>	Poziționare la sfârșit de fișier



# Poziționarea într-un fișier

---

- ***seekp(0, ios::beg)***
  - Deplasare cu 0 caractere relativ față de începutul fișierului
- ***seekp(-2, ios::end)***
  - Deplasare cu 2 caractere înainte de sfârșitul fișierului
- ***seekp(10, ios::cur)***
  - Deplasare cu 10 caractere față de poziția curentă din fișier

# Formatarea intrărilor

---

- 2 metode
  - Cu ajutorul funcțiilor membre
  - Cu ajutorul manipulatorilor
  
- Bazată pe câteva *flag*-uri:
  - *Flag*-uri on/off – prin folosirea metodelor *setf/unsetf*
  - *Flag*-uri care lucrează în grup

# Formatarea intrărilor

---

## □ *Flag*-uri on/off

<b>Flag</b>	<b>Explicație</b>
<i>ios::skipws</i>	Ignorarea spațiilor
<i>ios::showbase</i>	Afișarea bazei de nume
<i>ios::showpoint</i>	Afișarea punctului zecimal
<i>ios::showpos</i>	Afișarea semnului
<i>ios::uppercase</i>	Generarea de litere mari
<i>ios::unitbuf</i>	Curățarea <i>buffer</i> -ului după încărcare

# Formatarea intrărilor

---

## □ *Flag*-uri grupate

- Recomandare – folosirea doar a unui flag dintr-un grup

<b>Flag</b>		<b>Explicație</b>
<i>ios::basefield</i>	<i>ios::dec</i>	Folosirea bazei 10
	<i>ios::hex</i>	Folosirea bazei 16
	<i>ios::oct</i>	Folosirea bazei 8
<i>ios::floatfield</i>	<i>ios::fixed</i>	Folosirea notației zecimale
	<i>ios::scientific</i>	Folosirea notației științifice
<i>ios::adjustfield</i>	<i>ios::right</i>	Ajustarea ieșirii la dreapta
	<i>ios::left</i>	Ajustarea ieșirii la stânga
	<i>ios::interval</i>	Ajustarea unui câmp prin inserarea de caractere la o anumită poziție internă

# Formatarea intrărilor

---

- Get/set pentru dimensiune
  - *streamsize* **widht**()
    - Returnează numărul minim de caractere care trebuie afișate pentru o anumită reprezentare
  - *streamsize* **widht**(*streamsize wide*)
    - Setează o nouă dimensiune
  
- Get/set pentru caracterul de umplere
  - *char* **fill**()
    - Returnează caracterul de umplere
  - *char* **fill**(*char filch*)
    - Setează caracterul de umplere la noua valoare (*fillch*) și îl returnează
  
- Get/Set pentru precizia punctului zecimal
  - *streamsize* **precision**( )
    - Returnează precizia (nr de zecimale)
  - *streamsize* **precision**( *streamsize prec* )
    - Setează precizia la noua valoare *prec*

# Exemplu

---

- A se consulta exemplul din directorul *12/streams*, funcția:
  - *formatFlags()*;

# Manipulatori

---

- ❑ Funcții speciale pentru a fi folosite în conjuncție cu operatorii de inserare (<<) și extragere (>>) ai obiectelor de tip *stream*
- ❑ Echivalenți cu flag-urile și funcțiile anterioare

Funcție	Explicație
<i>ios::skipws()</i> <i>ios::noskipws()</i>	Ignoră spațiile
<i>ios::showbase()</i> <i>ios::noshowbase()</i>	Afișează baza numerică
<i>ios::showpoint()</i> <i>ios::noshowpoint()</i>	Afișează punctul zecimal
<i>ios::showpos()</i> <i>ios::noshowpos()</i>	Afișează semnul
<i>ios::uppercase()</i> <i>ios::nouppercase()</i>	Generează litere mari
<i>ios::unitbuf()</i> <i>ios::nounitbuf()</i>	Curăță <i>buffer</i> -ul după inserări

# Manipulatori

Funcție		Explanation
<i>ios::basefield</i>	<i>ios::dec()</i>	Utilizarea bazei 10
	<i>ios::hex()</i>	Utilizarea bazei 16
	<i>ios::oct()</i>	Utilizarea bazei 8
<i>ios::floatfield</i>	<i>ios::fixed()</i>	Utilizarea notației zecimale
	<i>ios::scientific()</i>	Utilizarea notației științifice
<i>ios::adjustfield</i>	<i>ios::right()</i>	Ajustarea ieșirii la dreapta
	<i>ios::left()</i>	Ajustarea ieșirii la stânga
	<i>ios::interval()</i>	Ajustarea câmpului prin inserarea unor caractere a o anumită poziție internă



# Manipulatori de I/O

---

- ❑ *ostream& function(istream &is)*
- ❑ *ostream& function(ostream &os)*

<b>Tip</b>	<b>Funcție</b>	<b>Explicație</b>
Input	<i>ws</i>	Extrage spațiile
Output	<i>endl</i>	Inserează linie nouă și curăță <i>buffer</i> -ul
	<i>ends</i>	Inserează caracterul nul
	<i>flush</i>	Curăță <i>stream buffer</i> -ul

# Manipulatori cu argumente (parametrizați)

---

- Necesită includerea fișierului *header* <iomanip>

Funcție	Explicație
<i>setiosflags(mask)</i>	Setează <i>flag</i> -urile de formatare specificate de parametrul <i>mask</i>
<i>resetiosflags(mask)</i>	De-setează <i>flag</i> -urile de formatare specificate de parametrul <i>mask</i>
<i>setbase(base)</i>	Setează <i>flag</i> -ul de formatare pentru bază la valoarea <i>base</i>
<i>setfill(c)</i>	Setează caracterul de umplere la valoarea <i>c</i>
<i>Setprecision(p)</i>	Setează precizia operațiilor de ieșire la valoarea <i>p</i>
<i>setw (n)</i>	Setează dimensiunea pentru următoarea operație de inserare la valoarea lui <i>n</i>

# Exemplu

---

- A se consulta exemplul din directorul *12/streams*, funcția:
  - *manipulators()*;