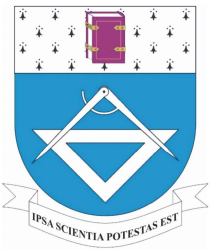


Universitatea Tehnică „Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare

Programarea Calculatoarelor II

Curs 1

Introducere. Recapitulare. Particularități ale limbajului C++.

Structura cursului

- Limbajul C++
  - Comparație între C++ și ANSI C, programarea orientată pe obiecte in C++
  - Abstractizarea datelor, constructori, destructori
  - Clase, membri unei clase, funcții prietene, clase prietene, pointeri la membri, supraîncărcarea operatorilor
  - Structuri de date în C++
  - Moștenire, clase derivate
  - Polimorfism, funcții virtuale pure, clase abstracte
  - Smart pointers
  - Șabloane de proiectare (design patterns) – exemple
  - Excepții în C++

Competențe obținute la finalul cursului

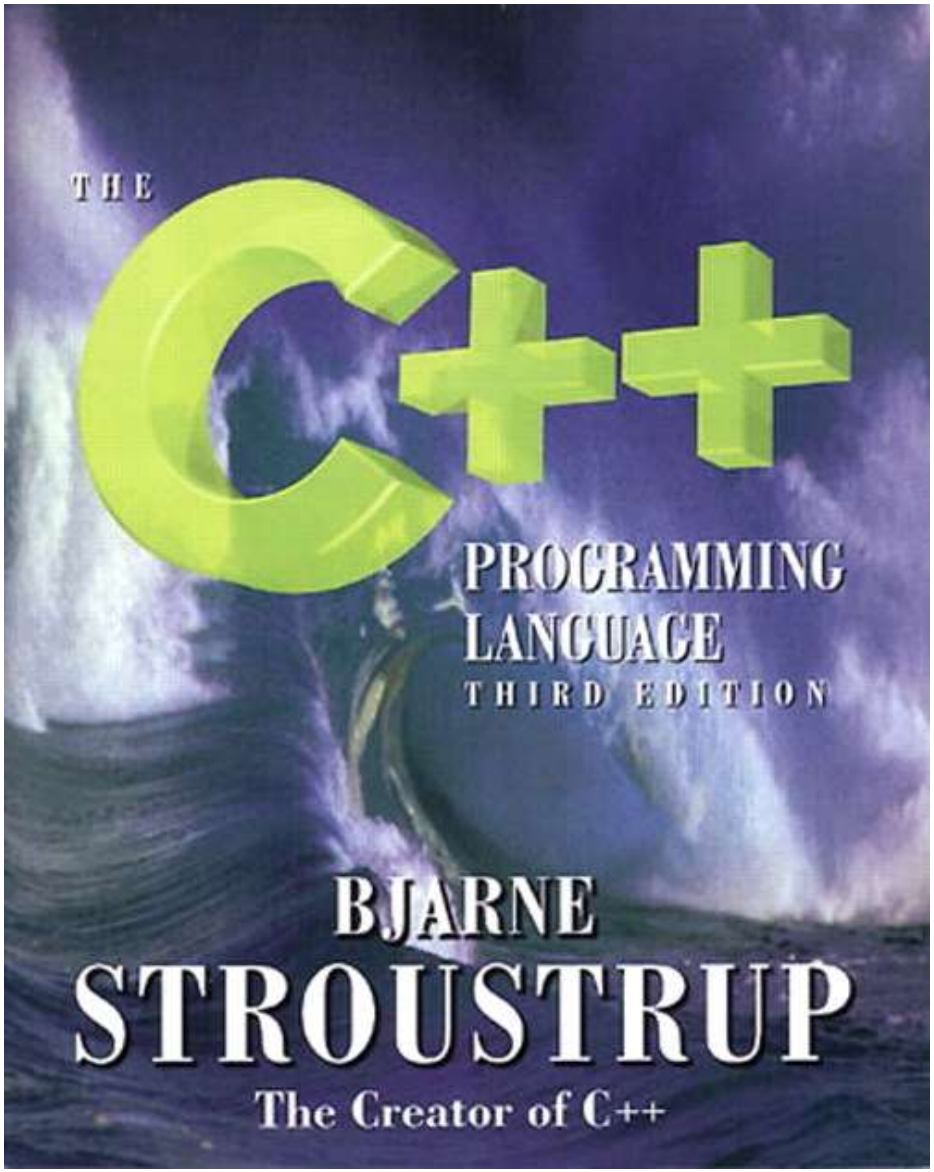
- Asimilarea tehnicii de programare pe obiecte
  - Poate fi aplicată în Java, C#, php, Python etc.
- Programarea orientată pe obiecte în C++

Calculul notei finale

Examen	**Test pe parcurs**	**50%**	**90% (minim 4.5)**
	Evaluare finală	**50% (minim 4.5)**	
Laborator			**10% (minim 5)**

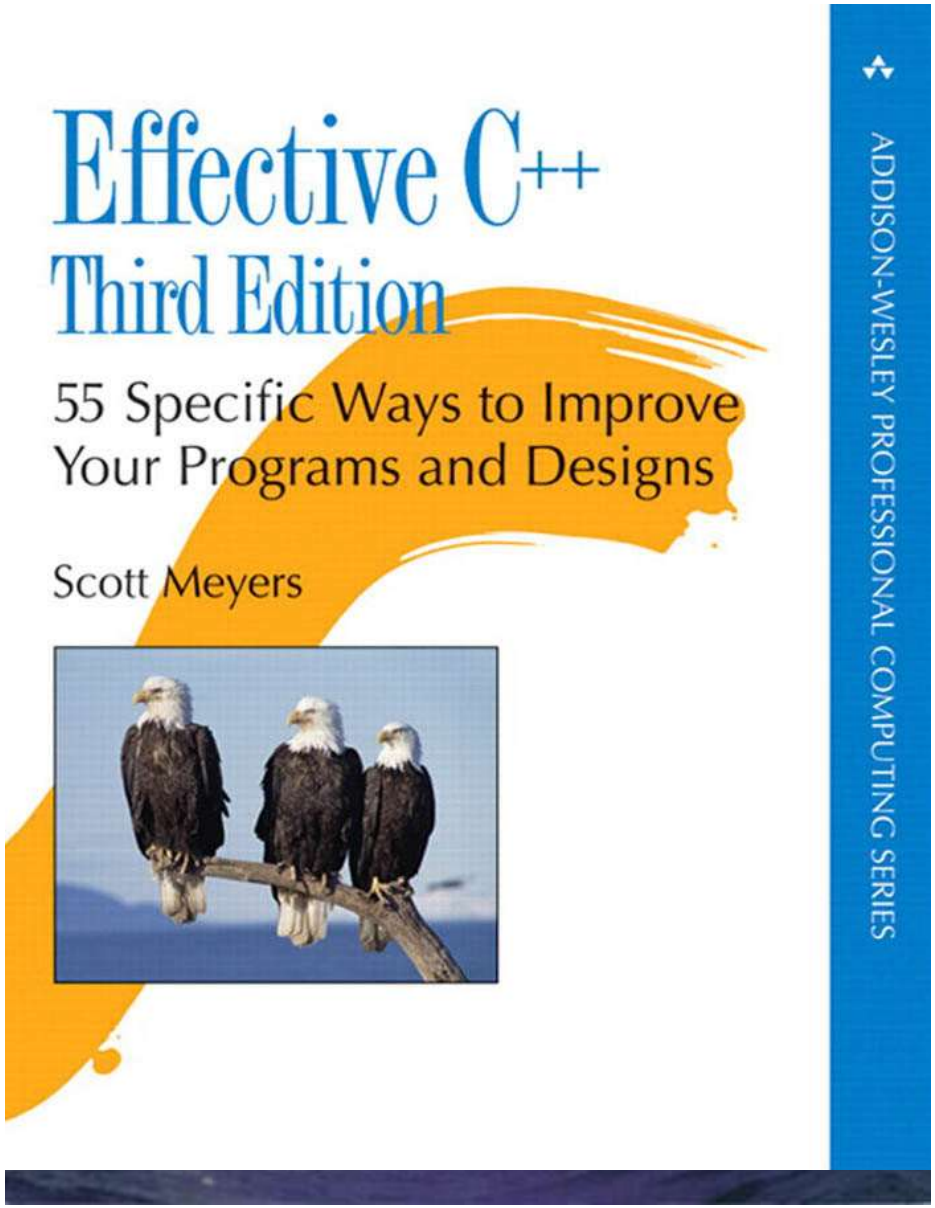
Bibliografie suplimentară

- The C++ programming language, third edition
  - Bjarne Stroustrup



Bibliografie suplimentară

- The C++ programming language, third edition
  - Bjarne Stroustrup
- Effective C++, third edition
  - Scott Meyers



Principalele tehnici de programare

- Activitatea de proiectare, codificare, testare și documentare a programelor se numește activitate de programare
  - **definirea generală a problemei**
    - se realizează schema logică conceptuală de rezolvare a problemei, se structurează datele de intrare/ieșire și se stabilesc algoritmii de calcul
  - **stabilirea logicii programelor**
    - descrierea algoritmilor în pseudocod sau utilizând scheme logice
  - **codificarea și testarea programelor**
    - se scrie textul sursă într-un limbaj, de obicei de nivel înalt și se testează programul cu date de test

Principalele tehnici de programare

- Activitatea de proiectare, codificare, testare și documentare a programelor se numește activitate de programare.
  - **implementarea și exploatarea programelor**
    - Programele sunt date în exploatare și se fac eventualele adaptări. Programele pot fi eventual împachetate în biblioteci utilizator;
  - **documentarea programelor**
    - *partea I-a*: documentația tehnică, care cuprinde: definirea problemei, schema conceptuală, logica problemei, textul sursă și datele de test.
    - *partea a II-a*: documentația de operare

Tehnici de programare

- Monolitică
- Procedurală
- Modulară
- Orientată pe obiecte

Introducere C++

- Numit inițial "C cu clase", redenumit în C++
- A apărut ca o îmbunătățire a limbajului C, la care s-au adăugat:
  - Clase
  - Funcții virtuale
  - Supraîncărcarea operatorilor
  - Moștenirea multiplă
  - Șabloane (Templates)
  - Altele

Tipuri de dată în C++

- char, unsigned char, signed char (1 octet)
- short [int], unsigned short [int] (2 octeți)
- int, unsigned [int] (4 octeți)
- long [int], unsigned long [int] (4 octeți)
- long long [int] (8 octeți)
- float (4 octeți)
- double, long double (8-10 octeți)
- **bool** (1 octet)

Operații de intrare – ieșire în C++

- `cin` pentru citirea de la tastatură
- `cout` pentru afișarea pe monitor
- Fișier header <iostream>
- `using namespace std;`
- Exemplu cin:

```
int c;
char j[40];
cin >> c;
cin >> j;
```

- Exemplu cout:

```
cout << j;
cout << "Nr citit este " << c << endl;
```

Tipul referință în C++

- Sinonim pentru o altă variabilă

```
int i = 8;
int &j = i;
j = 5;
cout << i << endl;
```

- Pe ecran se va afișa valoarea 5
- Transmiterea parametrilor funcțiilor prin:
  - Valoare (C, C++)
  - Referință (C++)
- O funcție poate returna o variabilă prin referință

Funcție interschimbare a doi întregi

C

```
void Schimba(int* a, int* b)
{
        int aux;
        aux = *a;
        *a = *b;
        *b = aux;
}

  int x=1,y=2;

  Schimba(&x,&y);
```

C++

```
void Schimba1(int &a, int &b)
{
        int aux;
        aux = a;
        a = b;
        b = aux;
}

int x=1,y=2;

Schimba1(x,y);
```

Afisare x afisare y

Mesajul afișat:

x=2 y=1

Funcții cu parametri impliciți

- Un parametru implicit este un parametru pentru care s-a specificat o valoare implicită
- Daca utilizatorul nu specifică o valoare pentru acest parametru la apelul funcției, valoarea implicită este utilizată
- Trebuie să se regasească în extrema dreaptă a listei parametrilor din antetul funcției

```
void afis(int a, int b = 0)
{
        cout << "a=" << a << " b=" << b <<endl;
}
```

Funcții cu parametri impliciți

- Pentru un apel de forma

```
int a=7,b=2;
afis(a,b);
```

- Se va afișa

```
a=7 b=2
```

- Pentru un apel

```
afis(a)
```

- Se va afișa

```
a=7 b=0
```