PROGRAMARE ORIENTATĂ OBIECT

Curs 1

Structura cursului
Competențe
Bibliografie

Principalele tehnici de programare
Limbajul C++, tipuri de date

Structura cursului

- Limbajul de programare C++
- Principii POO şi suportul oferit de limbajul C++
  - Abstractizare
  - Moştenire
  - Polimorfism
- STL

Competențe obținute la finalul cursului

- Asimilarea tehnicii de programare pe obiecte
- Limbajul de programare C++

Evaluare

- Activitate de laborator – 10%, minim nota 5 pentru a intra in examen
- Medie teste de laborator (2 teste anunţate) – 30%, nota obtinuta nu condiţionează intrarea în examen
- Examen (practic C++ fără Java) – 60%
- Proiect – 1 punct în plus la nota finală

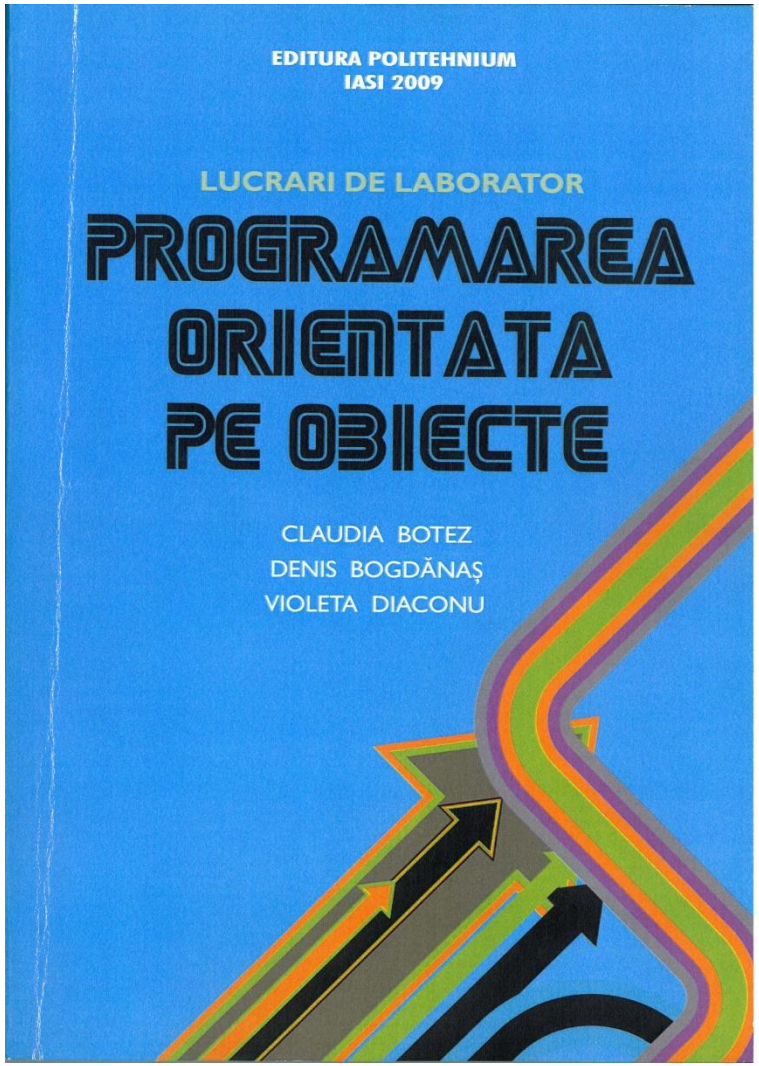
- Nota finală:
  - N = L x 0.1 + T x 0.3 + E x 0.6 + (E >= 8) x P x 0.1
- Verificare cunoştinţe teoretice în caz de incertitudine (rotunjire în plus sau în minus)

Bibliografie

- Programarea orientată pe obiecte, *Claudia Botez*

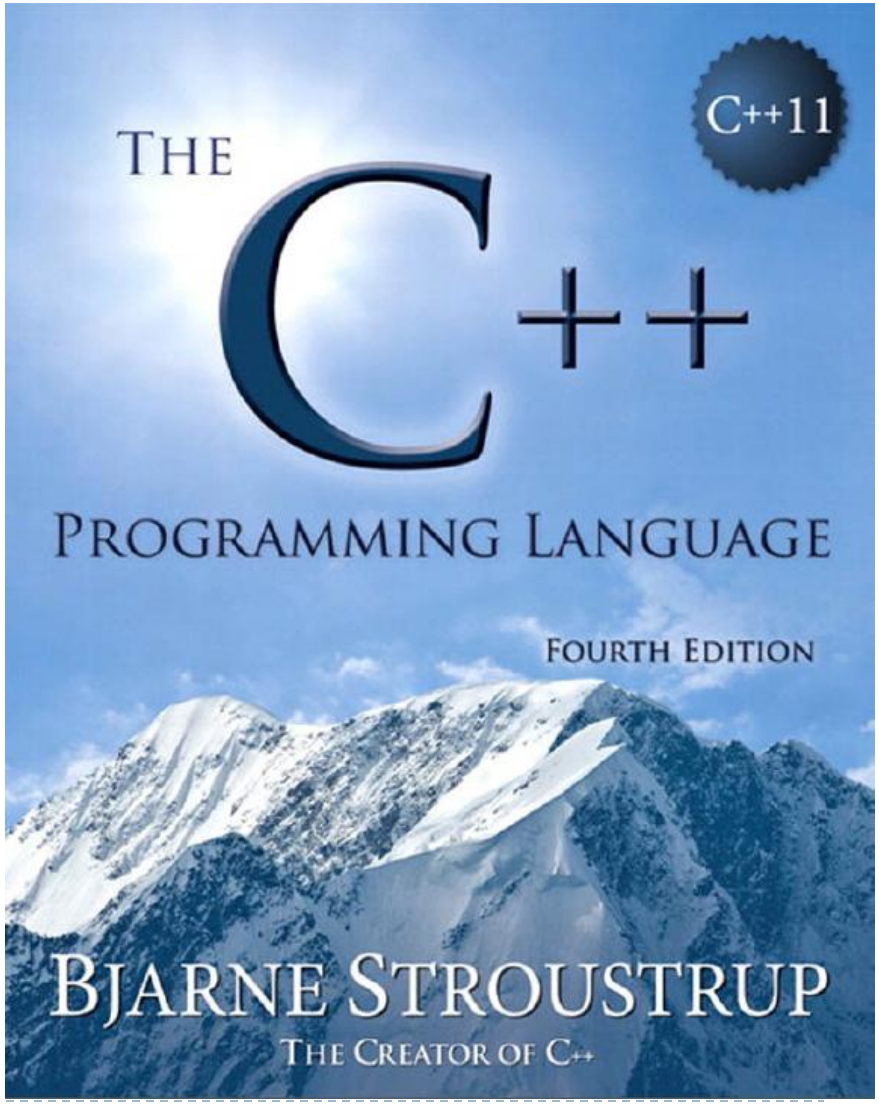
Bibliografie

- Programarea orientată pe obiecte, *Claudia Botez*
- Lucrări de laborator, Programarea orientată pe obiecte, *Claudia Botez*



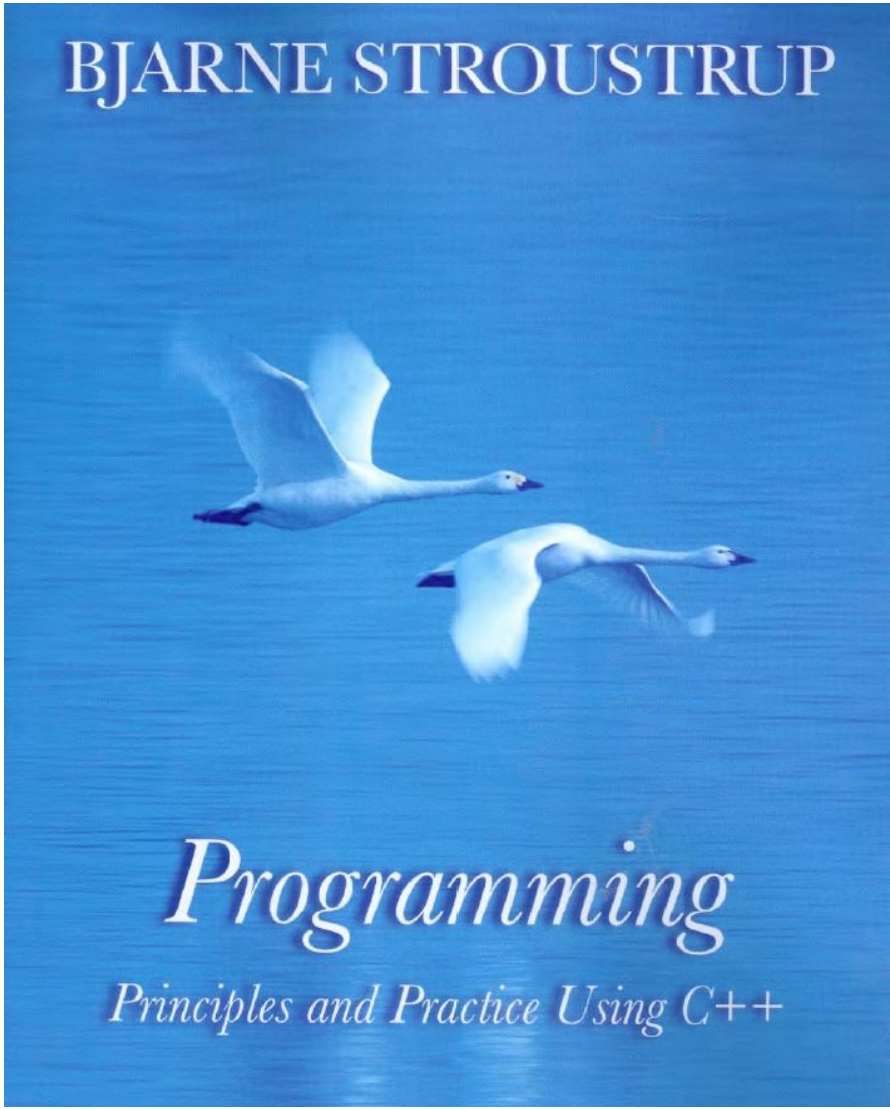
Bibliografie

- Programarea orientată pe obiecte, *Claudia Botez*
- Lucrări de laborator, Programarea orientată pe obiecte, *Claudia Botez*
- The C++ programming language fourth edition, *Bjarne Stroustrup*



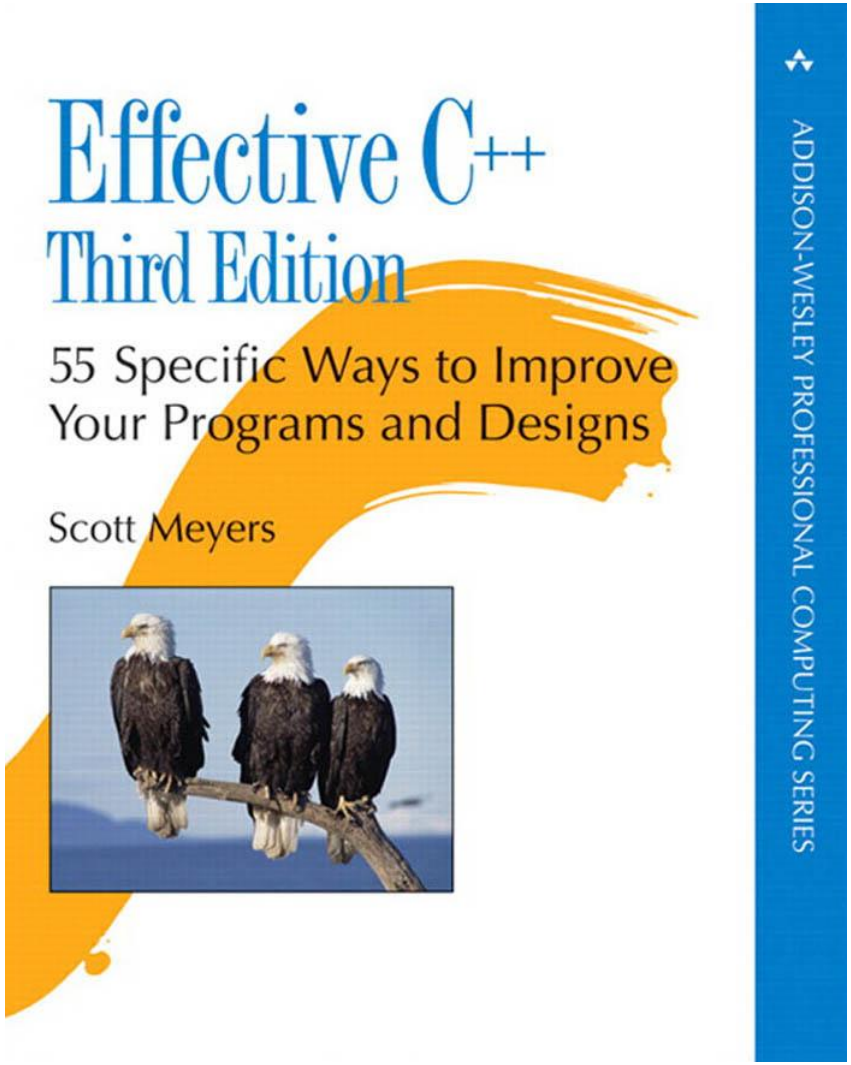
Bibliografie

- Programarea orientată pe obiecte, *Claudia Botez*
- Lucrări de laborator, Programarea orientată pe obiecte, *Claudia Botez*
- The C++ programming language fourth edition, *Bjarne Stroustrup*
- Programming principles and practice using C++, *Bjarne Stroustrup*



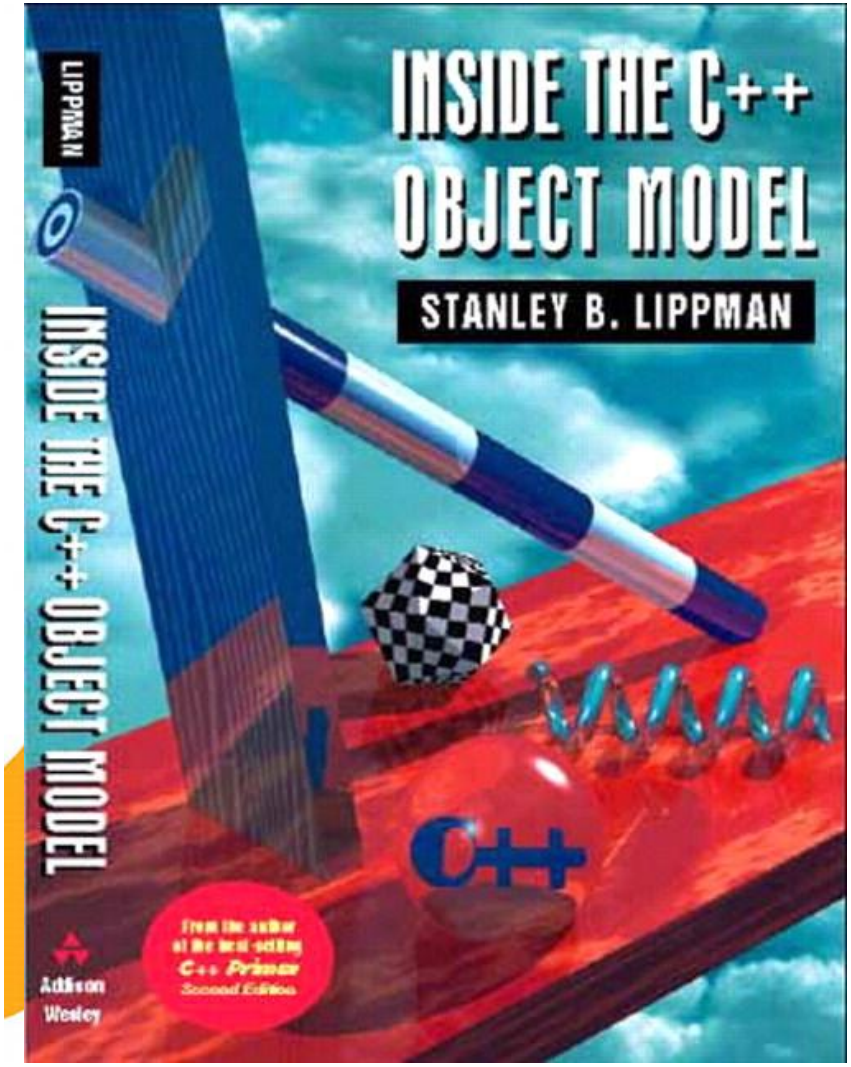
Bibliografie

- Programarea orientată pe obiecte, *Claudia Botez*
- Lucrări de laborator, Programarea orientată pe obiecte, *Claudia Botez*
- The C++ programming language fourth edition, *Bjarne Stroustrup*
- Programming principles and practice using C++, *Bjarne Stroustrup*
- Effective C++, third edition, *Scott Meyers*



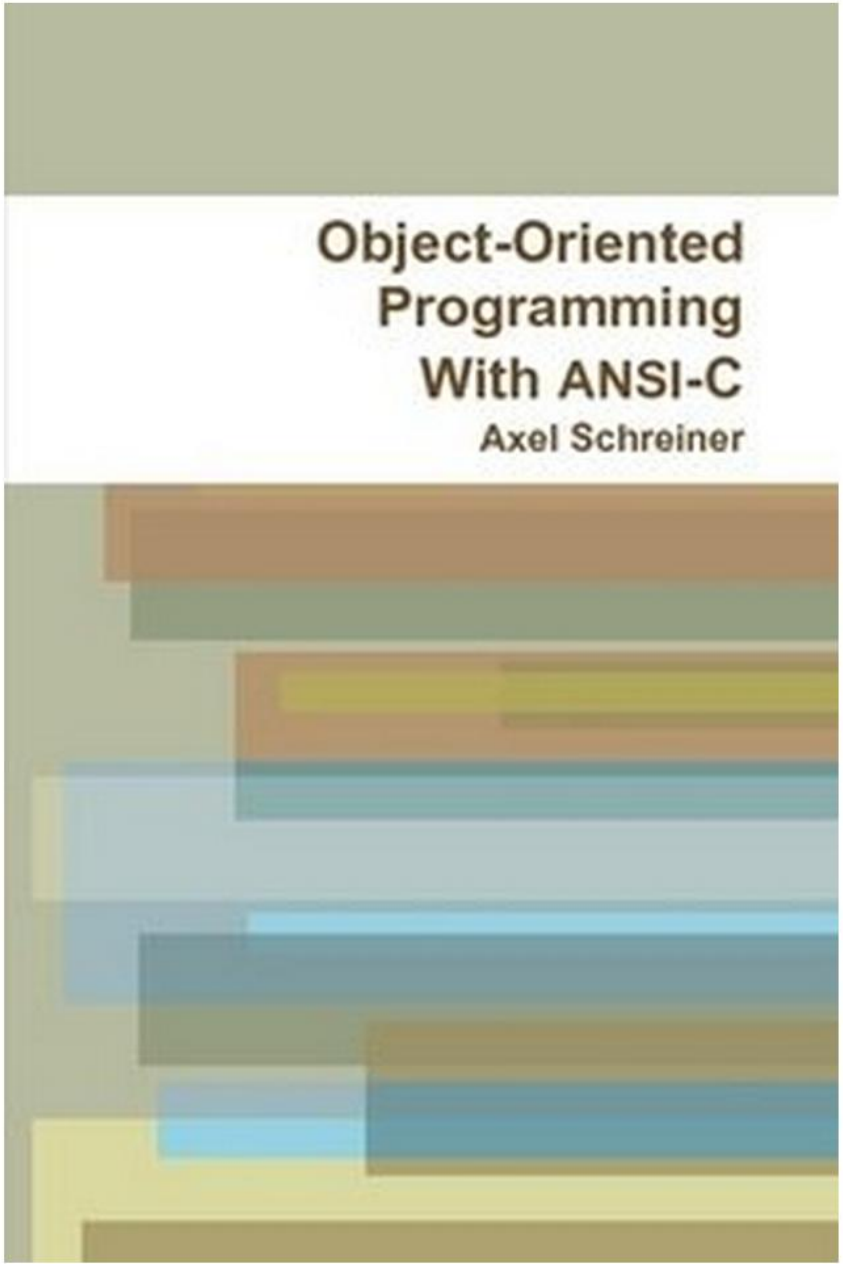
Bibliografie

- The C++ programming language fourth edition, *Bjarne Stroustrup*
- Programming principles and practice using C++, *Bjarne Stroustrup*
- Effective C++, third edition, *Scott Meyers*
- Inside the C++ object model, *Stanley B. Lippman*



Bibliografie

- Object-Oriented Programming with ANSI-C, *Axel Schreiner*



Principalele tehnici de programare

- Activitatea de proiectare, codificare, testare şi documentare a programelor se numeşte activitate de programare.
  - ***definirea generală a problemei***

    În această fază se realizează schema logică conceptuală de rezolvare a problemei, se structurează datele de intrare/ieşire şi se stabilesc algoritmii de calcul;
  - ***stabilirea logicii programelor***

    Această fază constă în descrierea algoritmilor în pseudocod sau utilizând scheme logice
  - ***codificarea şi testarea programelor***

    Acum, se scrie textul sursă într-un limbaj, de obicei de nivel înalt şi se testează programul cu date de test;

Principalele tehnici de programare

- ***implementarea şi exploatarea programelor***
  - Programele sunt date în exploatare şi se fac eventualele adaptări. Programele pot fi eventual împachetate în biblioteci utilizator;
- ***documentarea programelor***
  - *partea I-a:* documentaţia tehnică, care cuprinde: definirea problemei, schema conceptuală, logica problemei, textul sursă, şi datele de test.
  - *partea – II-a*: documentaţia de operare

Tehnici de programare

- Monolitică
- Procedurală
- Modulară
- Programării pe obiecte

Tehnica programarii monolitice

- Toată problema este rezolvată de la cap la coadă într-un singur program (funcţie)
  - Dezavantaje
    - Dificultate în depistarea erorilor (în programe mari)
    - Dificultate în întreţinere
    - Codul nu poate fi reutilizat

Tehnica programarii procedurale

- Nevoia de a reutiliza codul deja scris
- Secvenţa de instrucţiuni organizate în sensul rezolvării unei probleme se numeşte ***procedură*** sau ***subprogram*** sau ***subrutină***
  - Procedura nu întoarce nimic
  - Funcţia poate întoarce o valoare sau nu (void)
- Utilizare procedură/funcţie → proces de abstractizare realizat pe baza parametrilor
  - la definirea unei proceduri/funcţii se lucreaza cu parametri formali
  - la apelul unei proceduri/funcţii se lucrează cu parametri reali, efectivi sau actuali

Tehnica programarii procedurale

- Procesul de abstractizare prin intermediul parametrilor se numeşte abstractizare procedurală

- Utilizatorul nu trebuie sa ştie decât tipul parametrilor *a* şi *b* şi tipul valorii returnate

Tehnica programarii modulare

- Definiţia lui Bjarne Stroustrup, spune că un ***modul*** este un set de proceduri înrudite împreună cu datele pe care le manevrează.

- De multe ori datele unui modul sunt “ascunse”, adică accesul la ele este limitat, fiind protejate. De fapt accesul la aceste date este indirect, prin procedurile modulului (mai ales în programarea orientată pe obiecte).

Tehnica programarii modulare

- Fiecare modul se găseşte într-un fişier sursă, deci este **o unitate de compilare**. Prin urmare, modulele se compilează separat, rezultând din fiecare câte un modul obiect, care apoi se *leagă* într-un program executabil, de către editorul de legături.

Tehnica programarii modulare
Fişiere header (antet)

- Împărţirea unui program mare în mai multe module se face respectând condiţia ca orice modul să fie compilat independent. Acest lucru se întâmplă *dacă orice nume este utilizat numai după ce a fost declarat.*

- Legăturile între module se realizează prin intermediul funcţiilor şi al variabilelor globale, pentru care definiţia apare într-un fişier, iar în celelalte fişiere în care sunt folosite, ele sunt doar declarate.

Tehnica programarii modulare
Ce pot conține fişierele header

1) declaraţii de funcţii:

```
void ff(int);
extern void eroare (char  *);
```

2) declaraţii de variabile:

```
extern int i;
extern double f;
```

3) definiţii de tipuri, enumerări:

```
typedef int lungime;
struct complex {
    double re, im;
};
enum stare {OFF, ON};
```

Tehnica programarii modulare
Ce pot conține fişierele header

4) definiţii de constante, dar *nu* masive

```
const char ESC = ’\ x1B’;
```

5) directive preprocesor

```
#include <stdio.h>
#define MAX 100
```

6) definiţii de funcţii inline, specifice limbajului *C++*

```
inline int increment (int i)
{
    return ++i;
}
```

Tehnica programarii modulare
Ce ***nu*** pot conține fişierele header

1) definiţii de variabile globale

```
int i = 100;
double f;
```

2) definiţii de funcţii, care nu sunt inline:

```
void eroare (char *mesaj)
{
    printf ("\n %s\n", mesaj);
}
```

3) definiţii de masive (tablouri):

```
const int cifre [] = {0,1,2,3,4,5,6,7,8,9};
```

Utilizare

Un singur fişier antet

Mai multe fişiere antet

Tehnica programarii modulare
Compilarea condiționată

Fis.h

```
#ifndef __CONST__H
#define __CONST__H

extern int i;
void ff(int);
//..............continutul fisierului
#endif
```

Fis.h

```
#pragma once

extern int i;
void ff(int);
//..............continutul fisierului
```

Tipuri de date C++

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