Programarea calculatoarelor

Limbajul C

Cerinte

- Elaborarea a 2 proiecte (la seminar)
- Examen final - Test scris: intrebari de tip grila.
- Nota finala:
  - 60% testul scris;
  - 30% cele doua proiecte;
  - 10% punctaj din oficiu;
- Organizarea disciplinei: saptamanal 2 ore Curs, 2 ore laborator

Cuprinsul cursului

- Programul. Ciclul de dezvoltare a unui program. Algoritmi.
- Limbajul C
- Tipurile de date din C
- Operatii de intrare/iesire in C
- Expresii in C - Operanzi si operatori
- Realizarea structurilor fundamentale de control în limbajul C
- Tipuri dinamice de date. Pointeri
- Realizarea subprogramelor in limbajul C - Declararea si utilizarea functiilor
- Managementul fisierelor in C
- Funcții specifice de manipulare a șirurilor de caractere. Funcții specifice de manipulare a fișierelor text și binare
- Introducere in programarea orientate pe obiecte. Limbajul C++.

Bibliografie

- Kernighan & Ritchie: Limbajul C, Editura Teora, 2003
- Liviu Negrescu: Limbajele C si C++ pentru începători, volumul 1, partea I si II (Limbajul C), Editura Albastra, 2001
- H. Schildt - *Manual C complet* (Editura Teora, 1998 - traducere din limba engleză)
- Resurse online: www.cprogramming.com
- B.Brown, *C Programming*; http://www.eskimo.com/~scs/cclass/cclass.html

Programul

- **Programul** = prelucrarile ce trebuie efectuate de catre calculator asupra anumitor date
  - program executabil: cod masina interpretabil direct de calculator (procesor)
  - program sursa: ın limbaj inteligibil de programatorul uman
- **Algoritm**: o multime finita de reguli de calcul, descrise fara echivoc, care indica operatiile elementare si ordinea efectuarii lor scopul rezolvarii unei probleme intr-un timp finit.
- **Limbaj de programare**: limbaj folosit pentru scrierea de programe pentru calculatoare. Exemple: C, C++, Java, Python, etc.

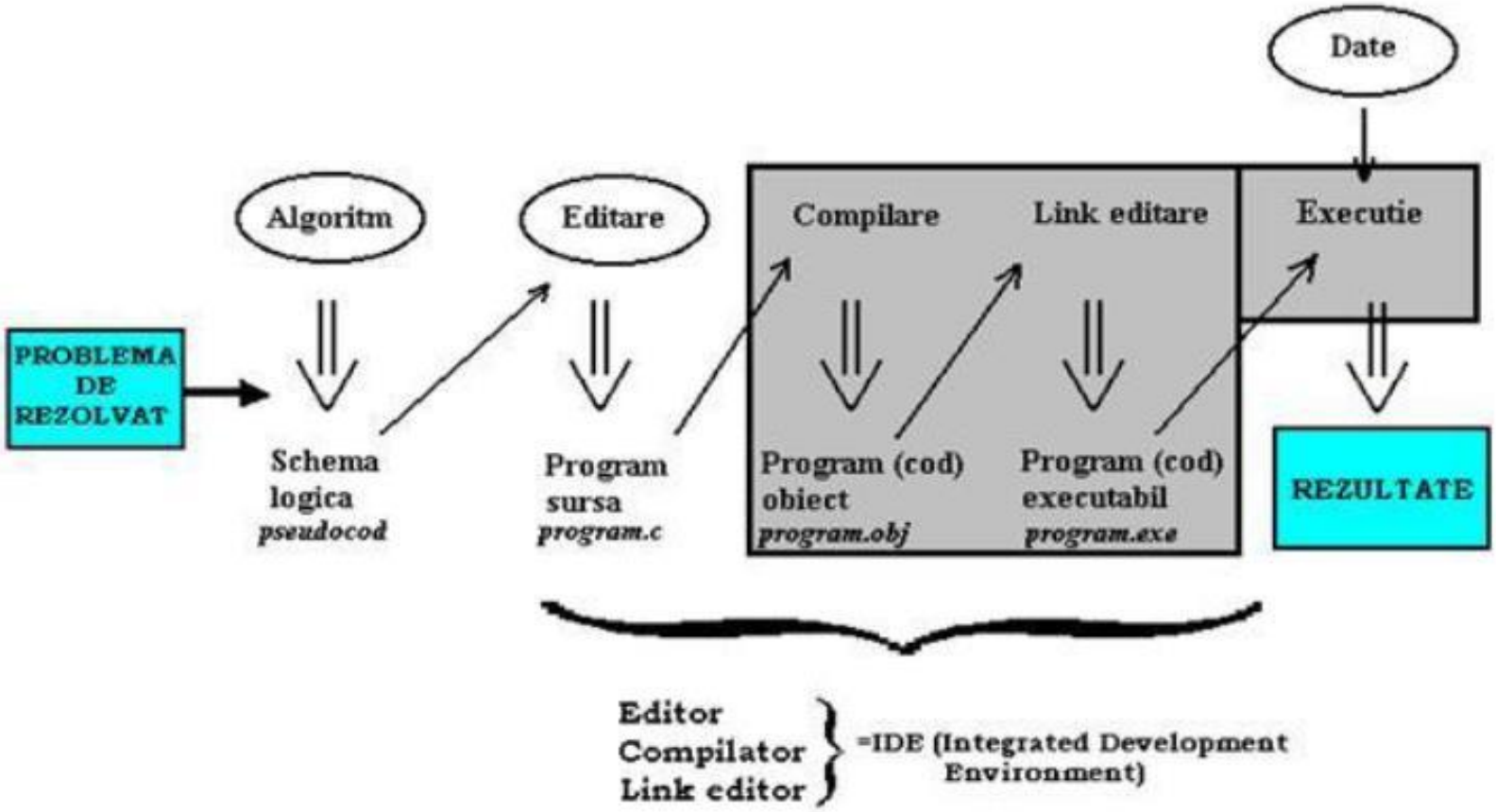
Programul

- Traducerea din format sursa in format executabil:
  - compilare si link-editare: anterior rularii programului (pt. C, C++, PASCAL)
  - interpretare: direct la rulare
- Compilator: traducatorul programelor sursa in cod obiect;
  - compilare: traducerea programului dintr-un limbaj evoluat in limbaj masina, adica translatarea instructiunilor programului intr-o forma caracteristica unui tip de calculator (codul obiect)
- Editor de legaturi (link-editor): uneste modulele obiect generate de compilator si produce codul executabil

Programul

- Un program generic:
  - citeste datele de intrare
  - efectueaza prelucrari (calcule) asupra lor
  - produce niste rezultate la iesire
- Etapele rezolvarii problemelor cu ajutorul calculatoarelor:
  - 1. Formularea problemei de rezolvat (specificatii de definire)
  - 2. Proiectarea solutiei (modelul matematic)
  - 3. Formularea algoritmului de rezolvare a problemei (date de intrare, date de iesire, tehnici de programare)
  - 4. Implementarea algoritmului folosind un limbaj de programare (Codare)
  - 5. Testarea corectitudinii programului si corectarea erorilor
  - 6. Mentenanta

Programul



Rezolvarea problemelor cu ajutorul calculatorului

Programul

- Testarea si corectarea erorilor
- Tipuri de erori:
  - De compilare (de sintaxa, semnalate de compilator)
  - Erori logice: datorita proiectarii gresite a algoritmilor (sunt detectate de regula prin obtinere unor rezultate neasteptate)
  - Erori de rulare (run-time) – sunt detectate in timpul executiei programului
- Procesul de depistare si corectare a erorilor = debugging.

Algoritmi

- **Algoritm**: secventa finita de operatii descrise fara echivoc care indica operatiile elementare si ordinea (fluxul) lor in scopul rezolvarii unei probleme intr-un timp finit.

Algoritmi

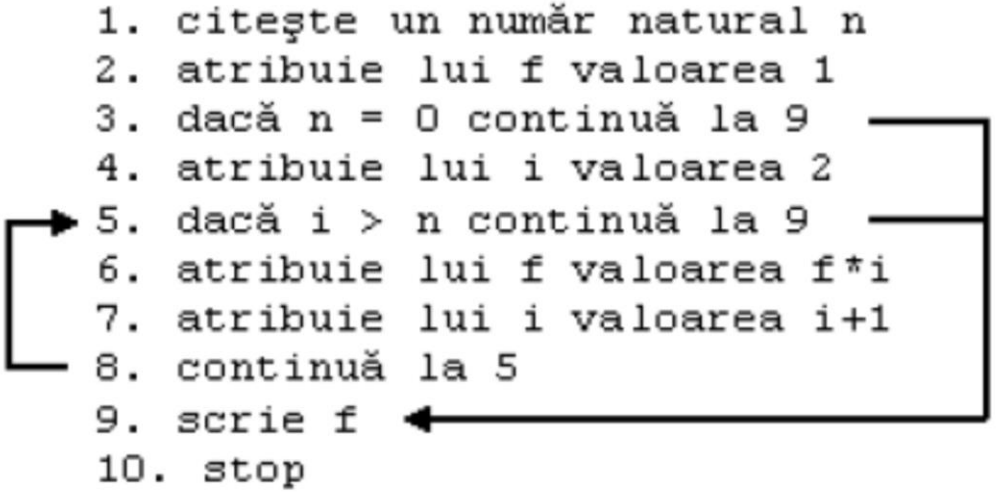
- Proprietati fundamentale ale algoritmilor:
  - **Generalitate** – trebuie sa ofere o metoda generala de rezolvare a unui anumit tip de probleme pentru date de intrare arbitrare.
  - **Precizie** – descrierea algoritmului trebuie facuta fara ambiguitati, intrstructiunile trebuie sa exprime operatii cunoscute calculatorului, care pot fi executate de procesor
  - **Finitudine** –obtinerea rezultatelor intr-un numar finit de pasi
  - **Eficienta** – trebuie sa foloseasca resurse hard cat mai putine si sa necesite un timp minim de executie
  - **Executabilitate** – algoritmul ca intreg si fiecare pas al sau trebuie sa poata fi executat.

Algoritmi

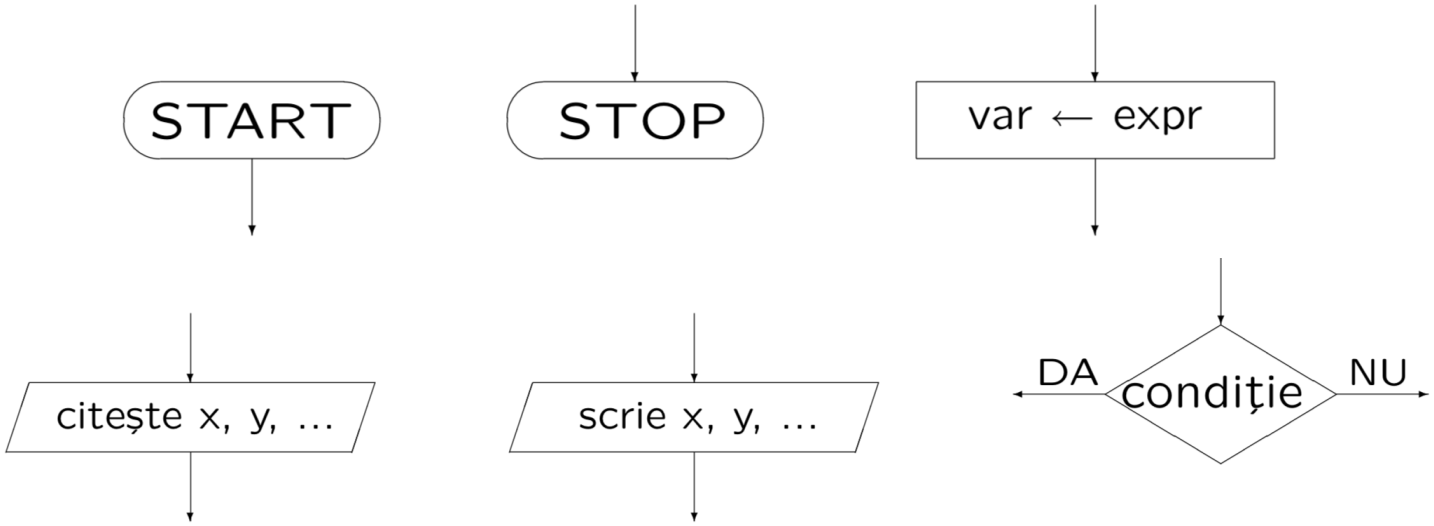
- Algoritmii se pot reprezenta prin:
  - Pseudocod (fara particularitatile unui limbaj dar precis)
  - Scheme logice (reprezentarea grafica a unui algoritm)

Algoritmi

- **Schema logica**: un graf format din cele 6 tipuri de instructiuni, avand un singur nod START si unul singur STOP.
- **Pseudocod**: exemplu – Factorialul unui numar

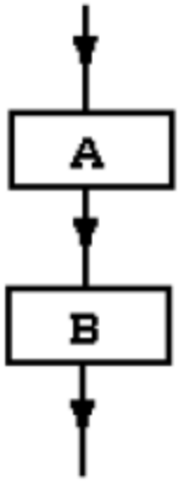


Algoritmi



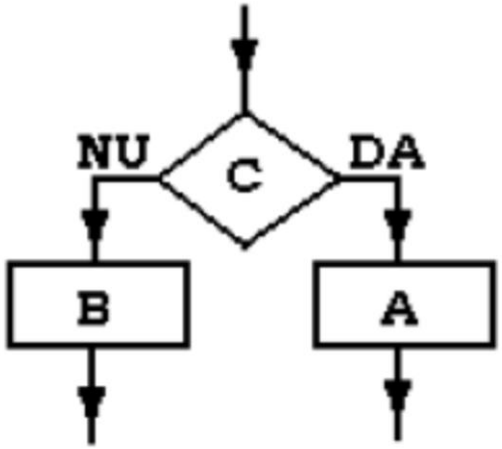
Structuri de control

- Structuri de control
  - permit o programare structurata
  - Orice algoritm poate fi reprezentat ca o combinatie a 3 structuri fundamantale de control:



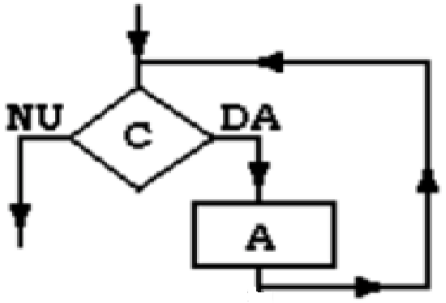
Structura secventiala

Structura alternativa

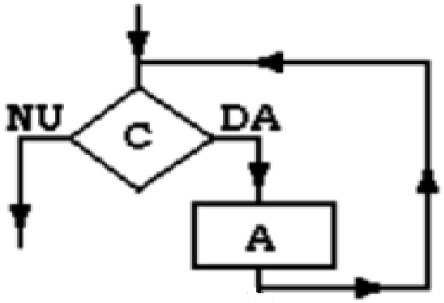


Structuri de control

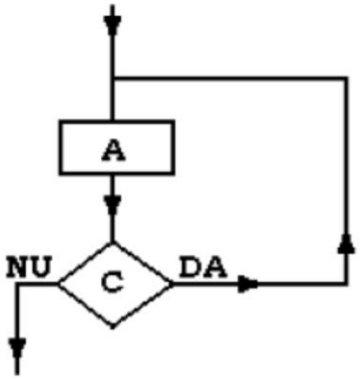
Structura repetitiva



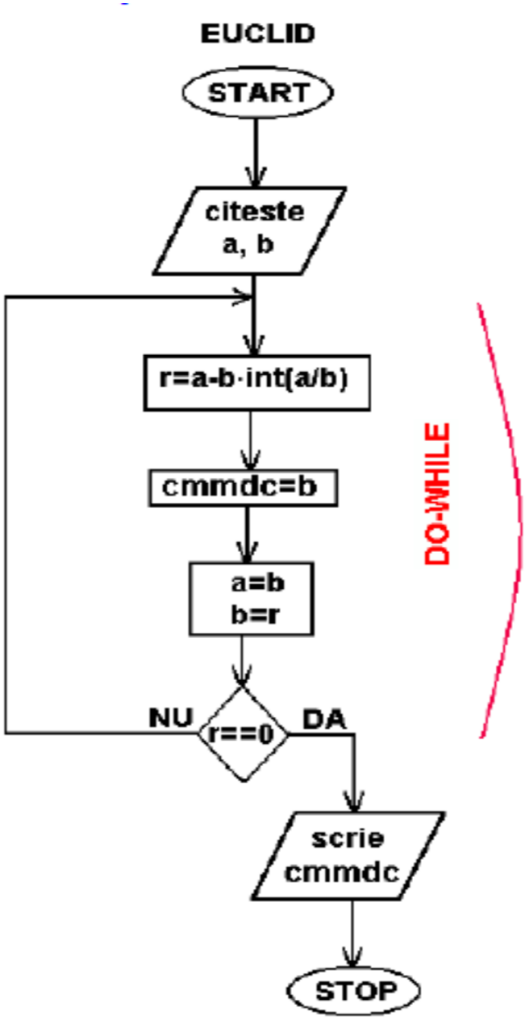
While-Do
(structura repetitiva cu test initial)

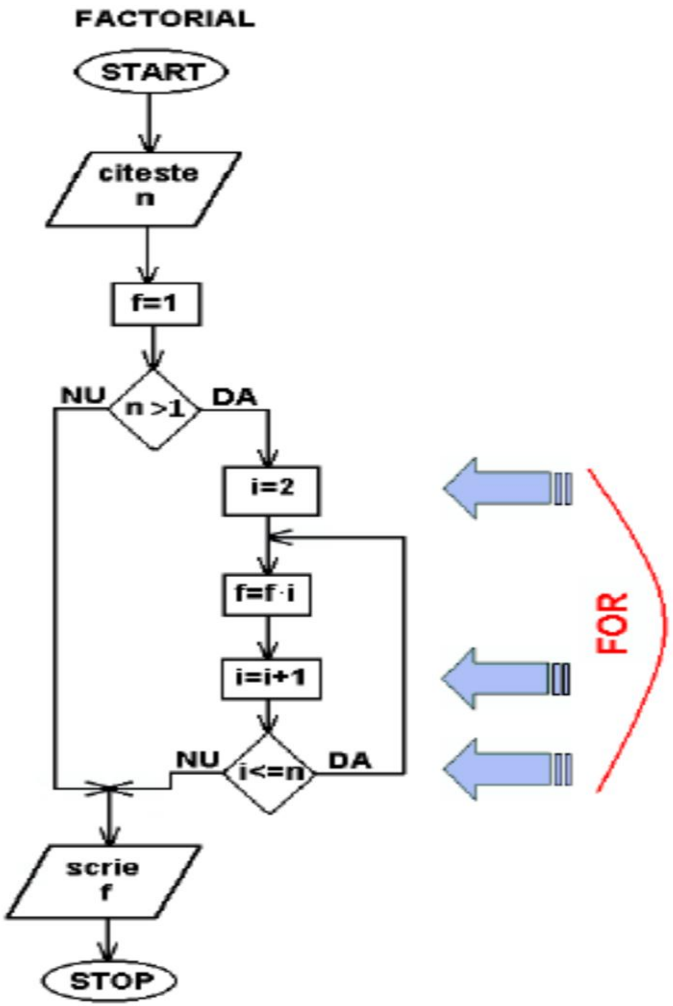


Do-While
Structura repetitiva cu test final



Structuri de control





Limbajul C

- Limbajul C
- Dezvoltat si implementat in 1972 la AT&T Bell Laboratories de Dennis Ritchie http://cm.bell-labs.com/cm/cs/who/dmr/chist.html
- Limbaj de programare structurata (blocuri, cicluri, functii)
- Necesitatea unui limbaj pentru programe de sistem (legatura stransa cu sistemul de operare UNIX dezvoltat la Bell Labs)
- C dezvoltat initial sub UNIX; in 1973, UNIX rescris in totalitate in C
- In 1988 limbajul a fost standardizat de ANSI

Limbajul C

- Limbaj de nivel mediu: ofera tipuri, operatii, instructiuni simple fara facilitatile complexe ale limbajelor de nivel (foarte) inalt
- Limbaj de programare structurat (functii, blocuri)
- Permite programarea la nivel scazut, apropiat de hardware
  - acces la reprezentarea binara a datelor
  - mare libertate in lucrul cu memoria
  - foarte folosit in programarea de sistem, interfata cu hardware
- Produce un cod eficient (compact in dimensiune, rapid la rulare) apropiat de eficienta limbajului de asamblare datorita caracteristicilor limbajului si maturitatii compilatoarelor
- Slab tipizat → necesita mare atentie in programare conversii implicite si explicite intre tipuri, char e tip intreg, etc.