Programarea calculatoarelor
Limbajul C

CURS 1

Ş.l. Carmen Odubăşteanu

Bibliografie:

- curs.cs.pub.ro – Programarea Calculatoarelor seria 1CC
- Negrescu L - Limbajele C şi C++ pentru începători, volumul 1: Limbajul C, Ed. Albastră, Cluj-Napoca, 2002
- Florian Moraru – Programarea Calculatoarelor
- Florian Moraru – Programarea Calculatoarelor (culegere)
- Brian W. Kernighan, Dennis M. Ritchie - The C Programming Language

- http://www.eskimo.com/~scs/cclass/notes/top.html
- http://www.eskimo.com/~scs/cclass/int/top.html
- http://cermics.enpc.fr/~ts/C/cref.html
- http://www.cs.bath.ac.uk/~pjw/NOTES/ansi_c/
- http://www.chris-lott.org/resources/cstyle/
- http://www.infoiasi.ro/fcs/absolvire4info.html

Cuprins

- Introducere
  - Algoritm şi obiectele acestuia
  - Limbaje de programare
  - Programare structurată
  - Scheme logice, pseudocod
  - Exemple
- Istoric limbaj C
- Elemente de bază ale limbajului C
  - Tipuri de date şi constante
  - Variabile şi operatori
  - Expresii
  - Funcții I/O
  - Instrucțiuni
- Directive de preprocesare
- Vectori

Cuprins

- Funcții
- Pointeri
- Vectori şi pointeri
- Funcții şi pointeri
- Şiruri de caractere
- Structuri
- Alocare dinamică
- Fişiere text şi fişiere binare

- Programe complexe. Compilări separate. Fişiere proiect.
- Convenții de programare

Obiective

Cursanții vor dobândi următoarele abilități:

- Să scrie, să compileze şi să ruleze un program simplu C utilizând mediul de dezvoltare DevCpp
- Să utilezeze corect elementele de bază ale limbajului C
- Să folosească tipurile structurate de date (vectori, matrici, structuri)
- Să proiecteze şi să implementeze programe modulare utilizând definirea de funcții proprii
- Să acceseze memoria folosind pointeri

Obiective

- Să acceseze şi să prelucreze date aflate in fişiere
- Să proiecteze şi să implementeze programe complexe sub formă de proiecte
- Să testeze şi să depaneze un program C
- Să folosească un stil de scriere a programelor cât mai eficient (comentarii, codificări)

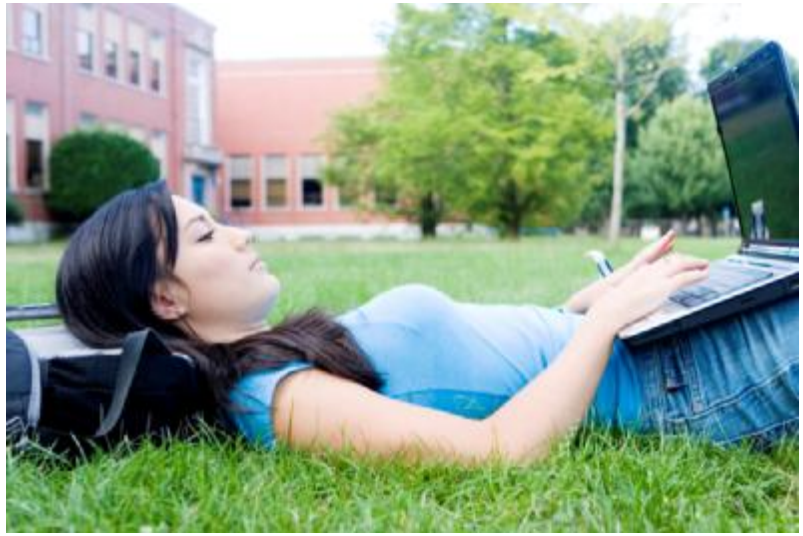
- Să treacă examenul de PC şi examenele de programare care vor urma în anii următori.

Uraaaaaa!

Obiective

Condiţiile pentru dobândirea acestor capabilităţi?

Dar beneficiile?



Nota finală

- Prezența laborator
- Activitate laborator
  - Lucrări neanunțate de maxim 30 min cu probleme asemănătoare sau chiar din laborator (laboratoarele anterioare + cel curent)
- Lucrări curs (?)
- Total din timpul semestrului: maxim 4p

- Examen final
  - condiție de intrare în examen: 50% din punctajul maxim din timpul semestrului

Teme de casă pentru cei care vor să lucreze mai mult - bonus (?)

Exemplu

$$F(x)=\begin{cases} x^2-2, & x<0 \\ 3, & x=0 \\ x+2, & x>0 \end{cases}$$

x = {-3, 0,1,7, 2.23, etc} – 100 valori

- Etape
  1. Elaborare algoritm
  2. Transpunere algoritm în limbaj de programare
  3. Rulare şi ... din nou etapa 1 dacă nu am obținut ce trebuia

Algoritm

- Succesiune de etape ce se poate aplica mecanic pentru rezolvarea unei clase de probleme
- Redactare
  - Scheme logice
  - Pseudocod
  - Mental – cine îşi permite?
- Cerințe
  - Claritate – fără ambiguități
  - Generalitate – pentru o întreagă clasă de probleme
  - Finitudine – furnizare rezultat în timp finit

Obs: O problemă poate avea mai mulți algoritmi de rezolvare – cel mai bun?

Obiecte cu care lucrează algoritmii

- Date
  - Intrare
  - Ieşire
- După tipul lor:
  - Întregi: 2, -4
  - Reale: 3.25
  - Logice: true şi false
  - Caracter: ‘y’
  - Şir de caractere: “ab23_c”
- Constante: date conținute în program fără a fi citite sau calculate - π

Obiecte cu care lucrează algoritmii

- Variabile: nume unic, conținut diferit
  - Nume
  - Tip
  - Valoarea la un moment dat
  - Locul în memorie (adresa)

  x1, x2, x3.... -> x

  F(x1), F(x2), F(x3), ... -> F
  - x este de tipul real, are valoarea 3 la un moment dat şi se află în memorie la adresa 0xFF32

Obiecte cu care lucrează algoritmii

- Expresii
  - Construite cu constante, variabile, operatori
  - De mai multe tipuri, ca şi variabilele: 3*x+7, x<y, etc
- Operații
  - Intrare: preluarea unei date de la un dispozitiv de intrare
  - Ieşire: trecerea unei date din memorie către un dispozitiv de ieşire
  - Atribuire: x=3; y=x; y=x+y
    - Se evaluează expresia din dreapta atribuirii
    - Valoarea obținută este atribuită variabilei din stânga, care îşi pierde vechea valoare
  - Decizie

Program

- Descriere precisă şi concisă a unui algoritm într-un anumit limbaj de programare

Limbaje de programare

- Limbaje de nivel coborât, dependente de calculator:
  - Limbaj maşină
  - Limbaj de asamblare
    - mnemonice pentru operațiuni
    - simboluri pentru adrese
  - greu, dar interesant!

Limbaje de programare

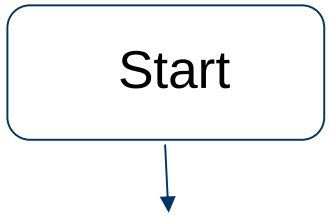
- Limbaje de nivel înalt, independente de structura calculatorului:
  - Fortran (FORmula TRANslation) – 1955, IBM, probleme tehnico-ştiinţifice
  - Cobol – 1959, probleme economice
  - Programare structurată – ‘70
  - Programare orientată pe obiecte – ‘80

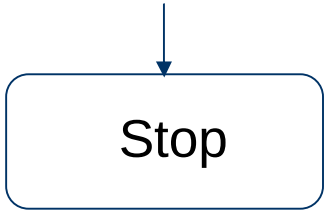
Programare structurată

- Dijkstra şi Hoare
- Programarea in care abordarea este top-down: descompunerea problemei complexe în subprobleme mai simple - modul
- Teorema de structură a lui Bohm şi Jacopini: orice algoritm poate fi compus din numai trei structuri de calcul:
  - structura secvențială - secvența;
  - structura alternativă - decizia;
  - structura repetitivă - ciclul.
  - o singură intrare şi o singură ieşire pentru fiecare

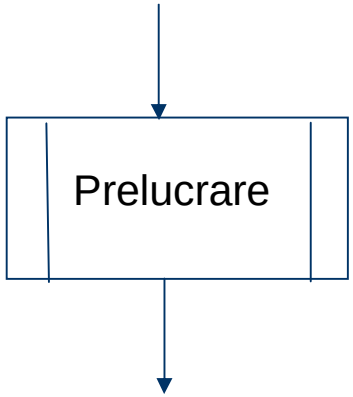
Operații auxiliare

- Start/Stop



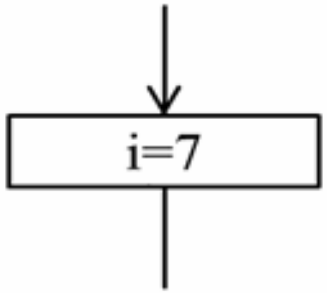


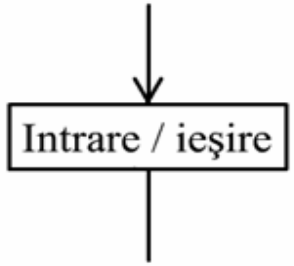
- Acțiuni nedetaliate



Secvența

- Atribuirea: operația prin care unei variabile i se atribuie o valoare.
- Operațiile de intrare/ieşire:
  - programatorul ia de la tastatură o valoare (intrarea)
  - afişează pe ecran o valoare (ieşirea)
- Pseudocod:
  - i=7;
  - citeste a;
  - scrie a;
- Exemplu: Citirea şi scrierea unei valori.





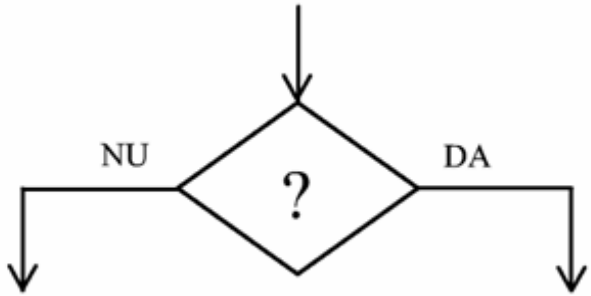
Decizia

- O întrebare ridicată de programator la un moment dat în program. In funcție de răspunsul la întrebare - care poate fi ori “Da”, ori “Nu” - programul se continuă pe una din ramuri.

- daca *conditie* adevarata

  instructiuni1;

  altfel instructiuni2;

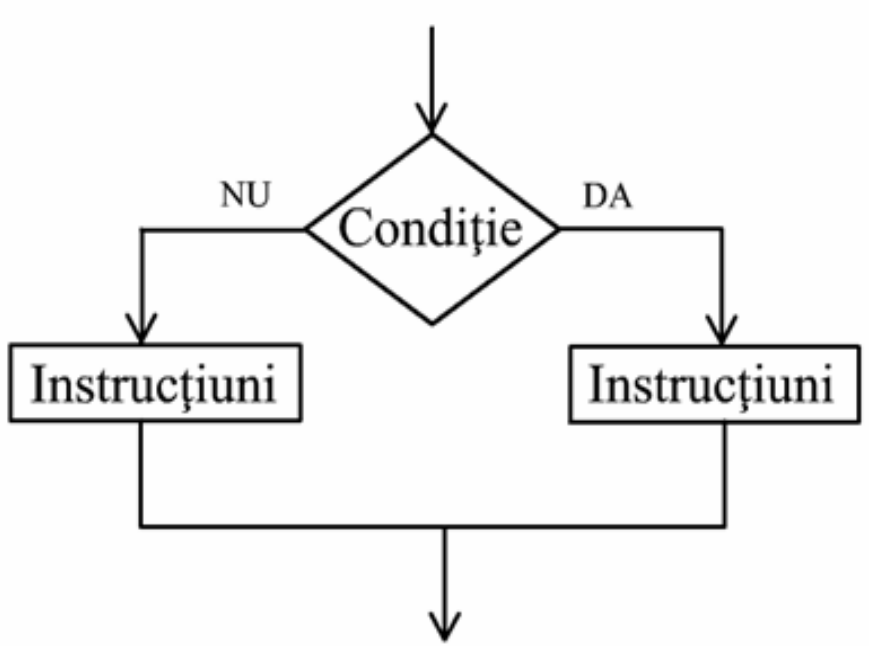


- Să se afişeze maximul

dintre două valori a şi b.

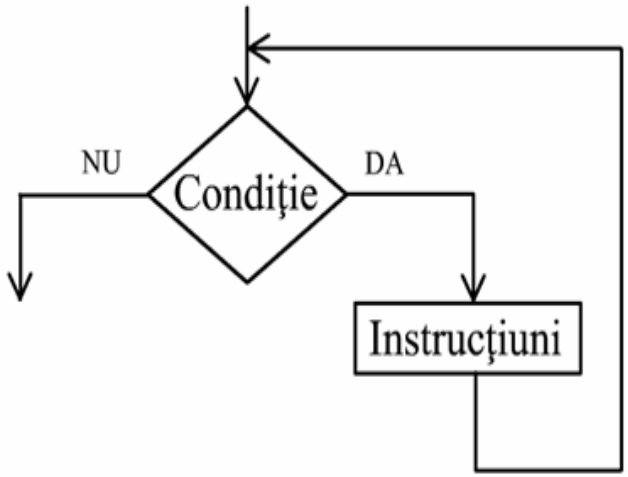
Structura alternativă

- Caz particular decizie
  - Condiție
  - Instrucțiunile care se execută dacă respectiva condiție este adevarată
  - Instrucțiunile care se execută dacă este falsă.
- Rezolvarea ecuației de grad 1: ax+b=0



Structura repetitivă cu conditie inițială

- O condiție, care se află la început
- Un bloc de instrucțiuni, care se execută dacă rezultatul evaluarii condiției este adevărat

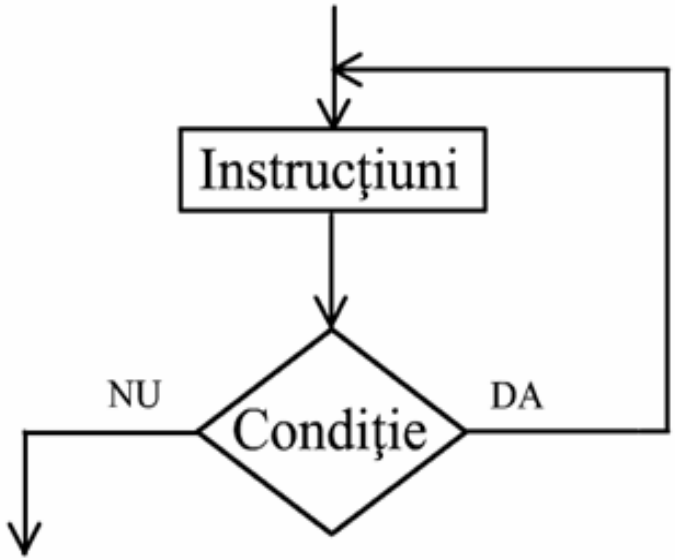


- atata timp cat *conditie* adevarata
  instructiuni

- Să se afişeze suma primelor n numere naturale, n citit de la tastatură.

Structura repetitivă cu condiție finală

- Bloc de instrucțiuni, apoi condiție.
- Obs: Blocul de instrucțiuni se execu minim o dată, spre deosebire de structura repetitivă cu test inițial, ur blocul de instrucțiuni era posibil să se execute deloc, dacă rezultatul evaluării condiției inițiale era fals.
- executa {
  instructiuni
  } atata timp cat *conditie* adevarata



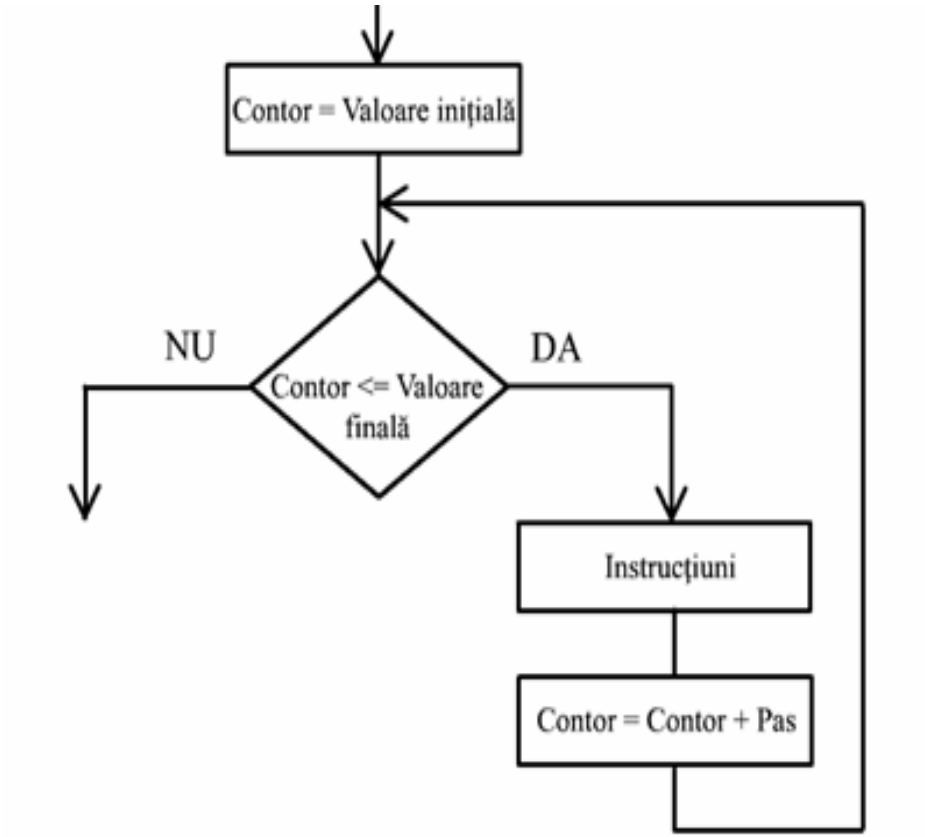
Structura repetitivă cu contor

- Caz particular al structurii de control cu test inițial.
- Utilizează o variabilă pe care o foloseşte ca un contor.
  - pleacă de la o valoare;
  - ajunge la o valoare;
  - înaintează cu un pas.

- pentru *contor* de la *val_initiala* la *val_finala* cu pasul *pas*

  instructiuni

- Suma primelor n numere naturale



Probleme propuse

1. Interschimbul valorilor a două variabile a şi b.
2. Rezolvarea ecuației de grad 2: $ax^2+bx+c=0$.
3. Să se afişeze în ordine crescătoare valorile a 3 variabile a, b şi c.
4. Să se calculeze şi să se afişeze suma: S=1+1*2+1*2*3+..+n!
5. Să se calculeze şi să se afişeze suma cifrelor unui număr natural n.
6. Să se calculeze şi să se afişeze inversul unui număr natural n.
7. Să se afişeze dacă un număr natural dat x este prim.
8. Să se afişeze primele n numere naturale prime.
9. Să se descompună în factori primi un număr dat n.
10. Să se afişeze toate numerele naturale mai mici decât 10000 care se pot descompune în două moduri diferite ca sumă de două cuburi.