Cursul 1
INTRODUCERE IN PROGRAMAREA PROCEDURALA (Limbajul C)

OBIECTIVE

- Prezentare generala
- Structura unui program
- Intrari/iesiri
- Variabile

PREZENTARE GENERALA

Conceptul de procedura, prezent in limbaje precum Pascal, Fortran, Basic sau C (sub diferite denumiri cum ar fi functie, procedura sau subrutina) permite segmentarea programului in module, numite proceduri. Filozofia transversala a paradigmei de programare procedurala (PP) aduce cu "divide et impera", fiecare procedura fiind mult mai simpla decat intregul program. Acest mod de programare faciliteaza dezvoltarea aplicatiilor in echipa, fiecare procedura putand fi implementata si testata independent de restul aplicatiei, fapt care faciliteaza, in plus, detectarea si corectarea erorilor de programare.

Mai mult, adoptand aceasta paradigma de programare, devine posibila obtinerea programelor fara utilizarea instructiunii goto, care complica enorm lizibilitatea, depanarea si mentenanta acestora. Claritatea si "independenta" procedurilor face, in plus, posibila obtinerea bibliotecilor de proceduri, reutilizabile in diverse aplicatii.

Un program in aceasta paradigma poate fi privit drept o colectie de date si proceduri ce se apeleaza intre ele si manipuland colectia de date. Conceptul de procedura permite structurarea acestor manevrari ale datelor, si deci reduce **intr-o oarecare masura** complexitatea acestor operatii, fara insa a reduce si complexitatea datelor. Unul din neajunsurile programarii procedurale il constituie tocmai efectul secundar al utilizarii neadecvate a variabilelor globale in cadrul unei aplicatii.

STRUCTURA UNUI PROGRAM

Un program in C, indiferent de marimea lui, consta din una sau mai multe functii care specifica operatiile efective ce trebuie efectuate. In limbajul C, functiile nu contin subfunctii (sau functii imbricate).

Functia **main()** joaca un rol foarte important in viata unui program in C pentru ca executia oricarui astfel de program incepe cu ea. Prin urmare un program C nu poate avea decat o singura functie **main()**.

Functia **main()** poate fi plasata oriunde in cadrul codului sursa al programului. Deoarece **main()**este o functie similara (ca structura si functionalitate) cu oricare alta functie din C, ea poate avea propriile sale tipuri de date locale, constante si variabile. In plus, functia **main()** poate sa intoarca o valoare, valoare care sa semnaleze o eventuala eroare in program.

Pentru a defini blocuri in program, in C se utilizeaza caracterele **{**, **}**, pentru inceputul, respectiv sfarsitul blocului. Acoladele sunt analoge lui BEGIN-END din Pascal. Fiecare instructiune dintr-un program C se incheie prin caracterul ";".

Programele scrise in C contin si directive de compilare. In acest limbaj exista urmatoarele directive de compilare: **#define, #undef, #include, #error, #if.**

Directiva **`#define`** este utilizata de programator pentru:

- a defini constante
- a inlocui cuvinte rezervate sau simboluri cu alti identificatori definiti de programator
- a crea identificatori pentru tipurile de date definite de programator, folosind tipurile de date standard
- a prescurta comenzi
- pentru a defini pseudofunctii.

Directiva **`#include`** permite includerea unor linii de cod sursa din alt fisier in fisierul curent, ca si cum s-ar introduce de la tastatura acele linii de cod in codul sursa al programului curent.

Sintaxa directivei **`#include`** este urmatoarea:

```
#include <fisier>
#include "nume_fisier"
```

Cele doua forme ale directivei difera prin modul in care se comanda compilatorului cautarea fisierului inclus. Prima forma cere compilatorului sa caute fisierul inclus intr-un catalog special, care contine numai asemenea fisiere, iar a doua forma extinde cautarea fisierului de inclus si in catalogul curent.

Un program in C poate sa contina si comentarii explicative. Comentariile care nu depasesc o linie sunt

precedate de caracterele "//". Pentru inserarea unui comentariu in cadrul codului sursa, acesta este incadrat de caracterele /*, */.

IESIRI

Functia **printf()** poate fi considerata o functie de conversie de format, cu scop general. Primul sau argument este un sir de caractere ce se va tipari, fiecare caracter % indicand argumentele *(al doilea, al treilea …)* ce se vor substitui, si forma in care se vor tipari. Fiecare constructie cu % in primul argument al lui **printf()** face pereche cu *al doilea*, *al treilea*, … argument, insa aceste perechi trebuie sa corespunda atat ca numar, cat si ca tip.

Functia **printf()** recunoaste:

- **"%f"** tipareste numarul ca flotant;
- **"%d"** intregi zecimali;
- **"%s"** pentru un sir de caractere;
- **"%c"** pentru un caracter;
- **"%%"** pentru semnul %.
- **'\t'** pentru tab
- **'\b'** pentru backspace
- **'\"'** pentru caracterul "
- **'\\'** pentru caracterul \

VARIABILE<home>

Prin **variabila** se intelege o locatie de memorie care poate gazdui un obiect ales dintr-o multime prestabilita de obiecte. O variabila apare in doua ipostaze:

- *locatie de memorie;*
- *pastratoare a unei valori.*

Variabila este caracterizata si prin modul de identificare a sa. Din acest punct de vedere inseamna ca o variabila va avea un *identificator*, ii va corespunde *o adresa* si va avea o *valoare.*

In limbajul C, toate variabilele trebuie declarate inainte de a fi folosite, de obicei la inceputul liniei, inaintea oricarei instructiuni executabile. O declaratie consta dintr-un *tip* si o *lista de variabile* care au acel tip.

In C exista urmatoarele tipuri de date predefinite:

- **int** (variabilele sunt intregi)

- **char** (de tip caracter)
- **double** (numar flotant in dubla precizie= real pe 64 biti)
- **float**(virgula mobila = real pe 32 biti)
- **void** (indica absenta oricarei valori)

Gama de tipuri de date devine mai flexibila prin adaugarea urmatorului set de modificari de tip:

- *signed*
- *unsigned*
- *short* (intreg scurt)
- *long* (intreg lung)

acestia afectand domeniul de valori si precizia datelor de tipul respectiv.

Exista, de asemenea, *tablouri*, *structuri*, *uniuni* si *clase* de astfel de tipuri de baza, *pointeri* la ele si functii care le returneaza.
Compilatorul face diferenta intre majuscule si minuscule atunci cand analizeaza variabilele si identificatorii.
Numele unei variabile t**rebuie sa inceapa cu o litera si sa contina alte litere, cifre si caracterul** "_".
In C, numele unei variabile poate avea orice lungime. Cand se declara o variabila intr-un program, este necesar ca acesteia sa i se asocieze un tip de date.
Limbajul C permite atribuirea de valori variabilelor, in momentul declararii acestora.

Sintaxa pentru declararea variabilelor este:

tip numeVariabila ;
tip numeVariabila=valoareinitiala;

```
int i;
double x=3.14;
```

Asa cum se preciza mai sus, in C se pot declara si liste de variabile de acelasi tip.

```
int j, i=2, k=3;
double x=3.14;
double y=2*x, z=4.5, a=45.7;
```

Valorile cu care sunt initializate variabilele pot contine alte variabile, definite anterior, sau constante.