7. FIŞIERE ŞI BAZE DE DATE DINAMICE

7.1. Fişiere

Programele prezentate anterior preiau informațiile de care au nevoie de la consolă, iar rezultatele (parțiale sau finale) sunt afişate pe ecran. Există însă numeroase situații, în care o parte a informațiilor prelucrate la o execuție a programului sunt necesare când acesta este executat din nou. La terminarea execuției unui program toate informațiile din memoria internă (legate de program) se pierd, deci la următoarea lansare în execuție datele de intrare trebuie introduse din nou. O soluție mai bună este înregistrarea datelor de intrare necesare pe un suport magnetic. Soluția este foarte avantajoasă dacă rezultatele execuției unui program urmează să fie prelucrate de un alt program. În acest caz este necesară salvarea rezultatelor într-o formă în care prelucrarea de către următorul program să fie comodă. Pentru aceasta programatorul îşi organizează informațiile într-un ***fişier***. Fişierele sunt memorate pe hard disc sau pe alt mediu extern. Fiecare fişier are un ***nume*** - un set de litere, numere şi simboluri permise, atribuit fişierului pentru a-l distinge de celelalte fişiere din director sau de pe disc. Numele de fişier este "mânerul" folosit de utilizatorul calculatorului pentru a salva sau a solicita blocuri de informații. Atât fişierele de program cât şi cele de date au câte un nume, iar uneori şi câte o extensie care identifică mai amănunțit tipul şi scopul fişierului. Convențiile de denumire, cum ar fi lungimea maximă şi caracterele permise ale unui nume de fişier, variază de la un sistem de operare la altul. În sistemul de operare DOS fiecare fişier are un nume format din 1-8 caractere şi o extensie de maxim 3 caractere.

PROLOG permite utilizarea hard discului sau a unităților de floppy disc pentru transferul de informații. Un fişier are, în afara numelui sub care este recunoscut de sistemul de operare, un ***nume simbolic***. Există patru nume simbolice predefinite prezentate în tabelul următor.

Nume simbolic	Dispozitiv	Input/Output

keyboard	tastatură	Input
screen	display	Output
printer	imprimantă (portul paralel)	Output
com1	portul serial	Input/Output

Pentru celelalte fişiere, numele simbolic este dat de programator. Acest nume este un şir de caractere alfabetice, cifre sau caracterul underline, primul caracter fiind literă mică. El trebuie declarat în secţiunea domains utilizând tipul file, sub forma:

file = nume_simbolic

Numele simbolice predefinite nu se trec în secţiunea domains. În interiorul programului ne putem referi la fişier prin intermediul numelui simbolic. Ordinea de executare a operaţiilor pentru un fişier este următoarea:

- deschiderea fişierului (pentru citire, scriere, adăugare, modificare)
- redirectarea fluxului de informaţii
- efectuarea prelucrărilor din fişier
- închiderea fişierului
- redirectarea către tastatură sau display

Pentru deschiderea unui fişier se pot folosi următoarele predicate:

- ***openwrite(NumeSimbolic, NumeFisier)***: dacă fişierul cu nume extern NumeFişier nu există, îl creează şi îl deschide pentru scriere; şterge conţinutul fişierul dacă a fost creat anterior.
- ***openread(NumeSimbolic, NumeFisier)***: deschide fişierul cu nume extern NumeFişier pentru citire.
- ***openmodify(NumeSimbolic, NumeFisier)***: deschide fişierul cu nume extern NumeFişier atât pentru citire cât şi pentru scriere.
- ***openappend(NumeSimbolic, NumeFisier)***: deschide fişierul cu nume extern NumeFişier pentru operaţia de adăugare la sfârşitul fişierului.

Pentru a specifica tipul fişierului (text sau binar) se poate folosi predicatul

filemode(NumeSimbolic, Mod)

Mod = 0 specifică modul text, iar Mod = 1 modul binar. Numărul maxim de fişiere deschise simultan este limitat de sistemul de operare.

Pentru redirectarea fluxului de informaţii există următoarele predicate predefinite:

▪ ***readdevice(NumeSimbolic)***: redirectează intrarea către fişierul identificat prin NumeSimbolic. Fişierul implicit de intrare este keyboard.

▪ ***writedevice(NumeSimbolic)***: redirectează ieşirea către fişierul identificat prin NumeSimbolic. Fişierul implicit de ieşire este screen.

Pentru prelucrarea fişierului, în afară de predicatele de intrare/ieşire (***write***, ***writef***, ***read***, ***readln***, ***readreal***, ***readint***) există următorele predicate predefinite

▪ ***filepos(NumeSimbolic, Poziţie, Mod)*** : se accesează o anumită înregistrare din fişier

Parametrul Mod indică punctul în raport cu care se consideră poziția înregistrării:

Dacă Mod este

0 numărarea se face pornind de la începutul fişierului

1 numărarea se face pornind de la poziția curentă

2 numărarea se face pornind de la sfârşitul fişierului înapoi

▪ ***eof(NumeSimbolic)*** : testează sfârşitul de fişier

Închiderea unui fişier se realizează cu ajutorul predicatului predefinit

closefile(NumeSimbolic)

Predicatul

flush(NumeSimbolic)

forțează transferul de date din buffer către fişier (fişierul rămâne deschis)

Următoarele predicate permit lucru cu fişiere sub sistemul de operare DOS:

- ***existfile(NumeFişier)***: testează existența fişierului
- ***renamefile(NumeFişierVechi, NumeFişierNou)***: schimbă numele fişierului
- ***disk(cale)***: defineşte calea curentă
- ***deletefile(NumeFişier)*** : şterge fişierul (din directorul curent)
- ***system("ComandăDOS")*** : execută o comandă DOS

Prezentăm în continuare câteva predicate predefinite pentru prelucrarea obiectelor de tip string:

- ***concat(String1, String2,String3)***: reuşeşte dacă şi numai dacă String3 se obține prin concatenarea lui String1 cu String2
- ***str_len(String,L)***: reuşeşte dacă L este numărul de caractere ale şirului String
- ***frontchar(String, C, Rest)***: reuşeşte dacă C este primul caracter al şirului String şi Rest reprezintă restul caracterelor.
- ***frontstr(N,StringIntrare, SN,SR)***: reuşeşte dacă SN reprezintă primele N caractere din StringIntrare şi SR restul; primele două argumente sunt de intrare, iar restul de ieşire

- ***fronttoken(StringIntrare,T,R)***: reuşeşte dacă T reprezintă partea de început a lui StringIntrare până la primul blank, şi R restul caracterelor
- ***upper_lower***, ***char_int***, ***str_char***, ***str_int***, ***str_real*** sunt predicate care realizează diverse tipuri de convertire

Exemplu. Se citeşte un text de la tastatură, linie cu linie.

Se cere:

- Să se memoreze textul într-un fişier. Numele fişierului va fi introdus de la tastatură. În cazul în care fişierul există, textul se va adăuga la sfârşitul fişierului.
- Să se afişeze conținutul fişierului în care s-a memorat textul.

```
domains
    file=text
predicates
     fer
     fer1
     fer2
     deschide_f(string)
     scrie_f(string)
     citeste_f(string)
     start
     nume_f(string)
     citeste(string)
     scrie(string)
clauses
    fer:-makewindow(1,111,36,"Nume fisier ?",1,1,22,78).
    fer1:-makewindow(2,113,37,"Scrie in fisier",2,2,10,77).
    fer2:-makewindow(3,113,37,"Citeste fisier",13,2,10,77).
    nume_f(X):-fer, cursor(10,30), write("Nume fisier = "), readln(X).
    deschide_f(X):-fer1,existfile(X), openappend(text,X),!.
    deschide_f(X):-fer1, openwrite(text,X).
    scrie_f(X):-deschide_f(X), writedevice(text),
```

```
 readln(Y),citeste(Y), closefile(text),writedevice(screen).
citeste(""):-flush(text).
citeste(Y):-write(Y),write("\n"),readln(X),citeste(X).
citeste_f(X):-fer2,openread(text,X),readdevice(text),
 readln(Y),scrie(Y), closefile(text),readdevice(keyboard).
scrie(Y):-eof(text),write(Y),!.
scrie(Y):-write(Y,"\n"),readln(X),scrie(X).
start:-nume_f(X), scrie_f(X),citeste_f(X),readchar(_).
```

7.2. Baze de date dinamice

Un program PROLOG este o bază de date în care datele sunt fapte şi reguli. Orice schimbare în datele programului impune actualizarea programului însuşi. În limbajele de programare tradiționale, actualizarea programului, în speță adăugarea sau ştergerea de date, precum şi modificarea fluxului controlului în program se fac de către programator. În PROLOG baza de date constituită din faptele şi regulile din program este "statică", în sensul că nu poate fi modificată decât între două execuții ale programului, dar există posibilitatea definirii de ***baze dinamice de date*** care pot fi actualizate automat în timpul execuției programului. O bază dinamică de date este o colecție de fapte. Programatorul poate să şi definească în program mai multe baze dinamice. Predicatele asociate bazei de date dinamice trebuie declarate în secțiunea database corespunzătoare sub forma:

database [-nume]

predicat1(tip_arg$_1$, tip_arg$_2$, …., tip_arg$_n$)

predicat2(tip_arg$_1$, tip_arg$_2$, …., tip_arg$_k$)

……

Un predicat declarat într-o bază de date dinamică poate fi utilizat oriunde în program, dar nu poate fi definit în program. Actualizarea bazei de date dinamice se face cu ajutorul predicatelor predefinite ***asserta(fapta)***, ***assertz(fapta)*** şi ***retract(fapta)***. Predicatele asserta şi assertz permit adăugarea faptelor într-o bază de date dinamică. Deosebirea dintre cele două constă în aceea că asserta adaugă la începutul bazei, iar assertz la sfârşitul bazei.

Conținutul unei baze dinamice se poate salva în vederea reutilizării ei. Salvarea se realizează cu ajutorul predicatului predefinit ***save(NumeFişier)*** unde NumeFişier este numele extern al fişierului ce va conține baza de fapte. Fişierul rezultat va fi o colecție se fapte, singura diferență față de un program fiind aceea că nu conține secțiuni. Orice alt program va putea să utilizeze această bază. Aceasta se realizează cu ajutorul predicatului predefinit ***consult(Nume)*** unde Nume este numele fişierului în care este memorată baza.

Ca exemplu considerăm programul de mai jos. În program se defineşte o bază dinamică de fapte de forma:

cititor(Nume, Prenume, Lista_cărți_împrumutate)

Cu ajutorul predicatului inregistreaza se introduc fapte de tipul

cititor(“Nume”, “Prenume”, [])

în baza dinamică (se înregistrează un cititor), iar cu ajutorul predicatului actualizare se actualizează această bază de fapte atunci când un cititor împrumută sau înapoiază o carte.

```
domains
        nume,prenume,carte=string
        lista_carti=carte*
database
        cititor(nume,prenume,lista_carti)
predicates
        id_cititor(nume,prenume)
        inregistreaza
        cititor_sol(nume,prenume,integer,carte)
        adauga_carte(carte,lista_carti,lista_carti)
        elimina_carte(carte,lista_carti,lista_carti)
        actual(nume,prenume,integer,carte)
        actualizare
        afis_carti(lista_carti)
        fer
        start
        baza
        meniu
```

```
        salvare
        selectie(char)
    clauses
        id_cititor(X,Y):-write("Nume= "),readln(X),write("Prenume="),readln(Y).
        inregistreaza:-id_cititor(X,Y), assertz(cititor(X,Y,[])),
                write("Continuati ?[d|n] "),readchar(Z),nl,Z='d',inregistreaza.
        inregistreaza:-salvare.
        cititor_sol(X,Y,I,C):-id_cititor(X,Y),write("Introduceti\n "),
                write("1 daca imprumuta o carte\n 0 daca inapoiaza o carte\n"),
                write("Optiune="),readint(K),K>=0,K<=1,I=K,
                write("Carte= "),readln(C).
        adauga_carte(C,X,[C|X]).
        elimina_carte(C,[C|X],X):-!.
        elimina_carte(C,[H|X],[H|Y]):-elimina_carte(C,X,Y).
        afis_carti([]):-nl,!.
        afis_carti([C|X]):-write(C,", "),afis_carti(X).
        actual(X,Y,I,C):-I=1,cititor(X,Y,L),adauga_carte(C,L,L1),
                retract(cititor(X,Y,L)),asserta(cititor(X,Y,L1)),afis_carti(L1).
        actual(X,Y,I,C):-I=0,cititor(X,Y,L),elimina_carte(C,L,L1),
                retract(cititor(X,Y,L)),asserta(cititor(X,Y,L1)),afis_carti(L1).
        actualizare:-cititor_sol(X,Y,I,C), actual(X,Y,I,C),
                write("Continuati ?[d|n] "),readchar(Z),nl,Z='d',actualizare.
        actualizare:-salvare.
        baza:-existfile("cititor.dat"),consult("cititor.dat"),!.
        baza.
        salvare:-system("del cititor.dat"),save("cititor.dat").
        fer:-makewindow(1,113,36,"Biblioteca",0,0,24,79).
        start:-baza, fer, meniu,removewindow.
        meniu:-clearwindow, cursor(10,20),write("i -> pentru inregistrare cititor"),
            cursor(11,20),write("a -> pentru actualizare lista carti"),
            cursor(12,20),write("e -> pentru iesire"),cursor(13,25),
```

```
        readchar(X),X<>'e',selectie(X),meniu.
    meniu.
selectie('i'):-clearwindow,inregistreaza.
selectie('a'):-clearwindow,actualizare.
selectie(X):-X<>'i',X<>'a',meniu.
```