CAPITOLUL 3

3. Noțiunea de variabilă şi tipuri de variabile.

O variabilă este un identificator (nume) dat după dorință de către utilizator unei zone din memorie cu care se va lucra (stoca sau modifica datele utile programului) pe parcursul scrierii codului sursă.

În limbajului C pot face asemenea referiri directe la o zonă de memorie (prin intermediul pointerilor) având totodată avantajul sintaxei facile a unui limbaj de nivel mediu.

Programul sau codul sursă este tradus prin intermediul compilatorului C într-un program executabil. În cadrul acestui proces se face identificarea dintre numele variabilei şi un anumit număr de locații de memorie. Tot în sarcina compilatorului rămâne şi generarea de instrucțiuni care să se ocupe de scrierea sau citirea în bloc a mai multor locații de o dată. O variabilă poate ocupa mai mult de o locație de memorie.
Caracteristicile variabilelor sunt :

1. Fiecare variabilă are un nume.
2. Fiecare variabilă aparține unui tip de dată.
3. Fiecare variabilă reține o valoare sau un set de valori utile pentru program. Aceste valori pot fi:
   - depuse în memorie prin inițializare în program
   - pot fi citite de la tastatură sau de la alt dispozitiv de intrare.

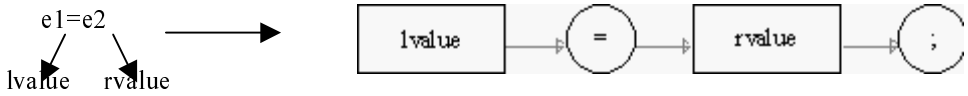
Legat de numele unei variabile se pot face câteva observații:

1. Numele unei variabile se alege de obicei astfel încât să descrie ce înseamnă pentru programator, din punct de vedere al pseudocodului într-o formă care poate fi prescurtată sau nu după preferință. De exemplu în cazul problemei din figura 1.2 numele unei variabile ce reține o notă putea fi : nota_1 sau n1 sau Nota_1. Ca observație limbajul C este “case senzitiv” (adică face distincție între literele mici şi mari folosite în scrierea codului sursă) deci NU SE CONSIDERĂ A FIIND ACELAŞI NUME DE VARIABILĂ ÎN CAZUL ÎN CARE ESTE SCRIS CU LITERE MARI SAU CU LITERE MICI. Mai concret nota_1 ≠ Nota_1≠NOTA_1.
2. Când se doreşte ca un nume de variabilă să fie compus din două cuvinte NU SE VA LĂSA SPAȚIU ÎNTRE ELE căci vor fi considerate două cuvinte separate, ele trebuiesc unite cu linie inferioară (underscore).
3. Numele de variabile suferă şi de următoarele restricții:
   - nu pot începe cu un număr (5j)
   - nu pot începe cu $ ($ abc)
   - nu pot fi cuvinte rezervate (int)
   - nu pot conține caractere speciale (bab*#*@l) excepție făcând “underscore” ex. bad_babe

3.1. Inițializarea variabilelor

În lipsa unei inițializări explicite, o variabilă locală va conține o valoare oarecare (aleatoare), numită în mod curent "garbage". Această valoare se găsea în memoria alocată respectivei variabile, la lansarea în execuția a programului

Inițializatorii expliciți se plasează după declaratorul la care se referă, separați de acesta prin semnul egal (de exemplu nr1=234).



Dacă obiectul inițializat este un agregat (un conglomerat de date recunoscut sub un nume comun), pentru inițializare se foloseşte o listă de expresii închisă în acolade unde la fiecare componentă corespunde câte o expresie.
Trebuie respectate următoarele reguli:

1. numărul de inițializatori nu poate fi mai mare decât numărul de componente de inițializat.
2. nu se pot inițializa decât obiecte sau tablouri cu dimensiune definită (adică se cunoaşte numărul maxim de valori de intrare ce pot fi date de utilizator pentru a fi procesate).
3. expresiile folosite pentru inițializare trebuie să fie constante în următoarele situații:
   - la inițializarea unui obiect de tip static (în C++, ne se mai aplică);
   - la inițializarea unui tablou, unei structuri sau unei uniuni.
4. nu se pot inițializa identificatori cu valabilitate în interiorul unui bloc, dar cu legare externă sau internă.
5. dacă o listă închisă între acolade are mai puține elemente decât componente are agregatul inițializat, restul componentelor se vor inițializa automat.

Obiectele de tip scalar (o serie de date de acelaşi tip, a căror definiție poate fi dată ca o înşiruire de date ce pot fi identificate prin poziția lor într-o înşiruire) se inițializează folosind o singură expresie, care poate fi eventual închisă în acolade. Uniunile (grupări de mai mulți biți de dimensiune divizibilă cu doi) se inițializează cu o singură expresie, corespunzând ca tip primului membru al uniunii.

Structurile (grupări de diverse date de tipuri diferite sub un nume comun) şi uniunile se pot inițializa cu o expresie de tip structură sau uniune corespunzătoare. Valoarea inițială a structurii sau uniunii va fi valoarea obiectului citat.

Prezența unei inițializări într-o declarație, indică o declarație definitorie (ce specifică cu ce va fi umplut respectivul spațiu de memorie). Mai jos este prezentat un exemplu.

```
int  zile [7] = { 1, 1, 1, 1, 1, 1, 1 } ;
char name1 [] = { "Costica" } ;
char name2 [] = "Dragan" ;
struct datep
   {
   int    i ; // nr. ani
   char   str [21] ; //nume
   double d ; //salar
   } vas = { 28, "Vasile", 1500000.0 } ;
```

Aceeaşi variabilă nu se poate iniţializa cu două valori, decât în cazul în care se renunţă la valoarea anterioară cu care a fost iniţializată.

Nu pot exista două declaraţii definitorii (care îi specifică conţinutul) într-un program.

3.2.Tipuri de variabile

Una din cele mai dificile probleme, care apar în cadrul învăţării unui limbaj de programare, este descrierea unei situaţii din lumea reală folosind elementele respectivului limbaj.

În afară de propunerea unui algoritm pentru a rezolva o problemă reală, programatorul mai este confruntat şi cu un alt aspect dificil: alegerea tipului de dată pentru o variabilă trebuie făcută în concordanţă cu valorile care le poate lua aceasta.

De aceea primul pas este să vedem ce tipuri fundamentale de date ne pune la dispoziţie limbajul C precum şi gama de valori în care evoluează acestea. O dată ştiute aceste lucruri putem începe să vedem care este legătura cu exemplele din viaţa reală.

Din analiza informaţiei care circulă curent între oameni rezultă că există, din punct de vedere al procesării, două tipuri de informaţie.

- informaţie obţinută prin măsurători asupra fenomenelor reale, care sunt exacte şi asupra cărora se fac prelucrări matematice de obicei;
- informaţie sub formă condensată care de obicei este recompusă asociativ în mintea umană, cum ar fi limbajul si informaţia scrisă.

De aici rezultă că într-un limbaj de programare avem nevoie de două tipuri de variabile, care să poată fi folosite pentru:

- stocarea unor valori numerice.
- stocarea unor caractere, informaţie scrisă.

Conform acestor necesităţi limbajul C a fost prevăzut cu următoarele tipuri fundamentale de date:

- **int** stochează numai numere întregi;
- **float, double** pentru numere zecimale , calcul în virgulă mobilă;
- **char** pentru a stoca codul unui caracter sau un număr întreg reprezentat pe maximum opt biţi incluzând şi semnul.

Observaţie : char[nr] va reţine un şir de nr caractere sau întregi.

Una dintre utilizările şirurilor de caractere ar fi mesajele de dialog cu utilizatorul unui program, sau mesajele de eroare. În acest caz nu ne interesează conţinutul din punct de vedere al valorii lor numerice ca atare.

Funcţie de necesităţile programatorului putem spune că există două clase de tip de date care pot fi folosite în cazul unui limbaj de nivel mediu sau înalt:

1. tipuri de date predefinite
2. tipuri de date definite de utilizator.