CAPITOLUL 6

6. Instrucțiuni de selecție (decizie).

6.1. Instrucțiune simplă de decizie

Fie următoarea problemă ce trebuie rezolvată prin intermediul unui program:

Să se realizeze un program care face automat calculul mediei de licență şi a mediei generale.

Programul primeşte trei medii de le examenele de licență şi nota primită la susținerea proiectului de stat. El trebuie să calculeze media între notele de la examenele de licență, rezultând o primă medie, care apoi se mediază cu nota de la proiectul de stat rezultând media generală.

Există o limitare asupra datelor de intrare care trebuie realizată de program şi anume dacă una din mediile de la examenul de licență este mai mică ca cinci să nu mai realizeze calculul mediilor finale şi să-l declare pe respectivul candidat respins.

O soluție de rezolvare a problemei la nivel de pseudocod arată ca mai jos:

```
include librăriile necesare lucrului în mod text
void main(void)
      {//început program
      declară variabilele n1,n2,n3,ns,m1,m2 de tip real;
      şterge ecranul;
      citeşte nota1 în variabila n1;
      citeşte nota2 în variabila n2;
      citeşte nota3 în variabila n3;
       dacă (n1>5) şi (n2>5) şi (n3>5) atunci
                        {
                  citeşte nota stat în variabila ns;
                        realizează media m1=(n1+n2+n3)/3;
                        realizează media m2=(m3+ns)/2;
                        afişează media de licența m1;
                        afişează media generală m2;
                        }
       altfel afişează "Candidat respins";
      aşteaptă o tastă;        }// sfârşit de program
```

Se observă că a fost nevoie de o ramificare la nivelul procesării datelor, ramificare care apare din nevoile utilizatorului. Este nevoie ca la un moment dat să hotărâm moduri multiple de prelucrare pentru acelaşi flux de date de intrare.

În limbajul C pentru a implementa acest tip de decizii a fost introdusă instrucțiunea if… cu sintaxa ca mai jos:

```
if (condiție=adevărată)                    dacă condiția este adevărată
    {                                                    execută    bloc
instrucțiuni1
     execută bloc instrucțiuni1      altfel
     }                                                   execută bloc instrucțiuni1
     else
     {
     execută bloc instrucțiuni2
     };
```

Sintaxa instrucțiunii este identică cu pseudocodul, bineînțeles că se foloseşte limba engleză.

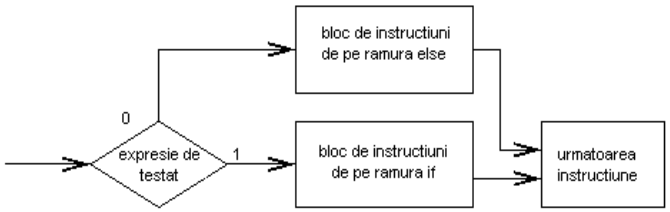

Fig. 6.1a. Reprezentarea prin schema logică a instrucțiunii de decizie

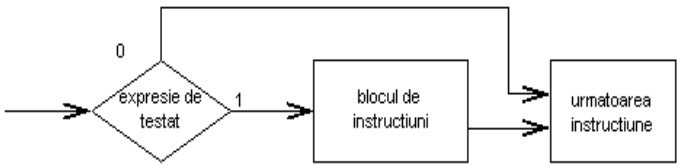

Fig. 6.1b. Reprezentarea prin schema logică a instrucțiunii de decizie simple

În figurile 6.1a şi 6.1b este prezentată reprezentarea la nivel de schemă logică a instrucțiunii de decizie. Spre deosebire de alte limbaje de programare, cum ar fi PASCAL de exemplu, limbajul C nu foloseşte variabile de tip logic pentru rezultatul evaluării unei expresii. Valorile comparate pot rezulta din valoarea returnată de o funcție, din cea stocată într-o variabilă sau direct dată în program.

Dacă în locul condiției, în urma evaluării expresiei rezultă un număr diferit de zero, acesta va fi considerat ca condiție îndeplinită şi deci se va executa blocul1 de instrucțiuni, dacă este zero condiția nu este satisfăcută atunci se va executa blocul2 de instrucțiuni. Acest lucru conferă un mare avantaj prin faptul că se pot introduce direct expresii matematice sau apeluri de funcții ce returnează rezultate numerice în zona de test a condiției.

Nu întotdeauna este nevoie de ambele părți ale instrucțiunii, de exemplu când se execută o operațiune numai dacă este îndeplinită o anumită condiție. De fapt structura logică de control ar trebui prevăzută şi cu o ramură de else dar acest lucru nu ar duce decât la supraîncărcarea codului, ceea ce îngreunează mult depanarea programului, rezultând sintaxa prezentată mai jos:

if (condiție=adevărată) *dacă condiția este adevărată atunci*
{ *execută bloc instrucțiuni*
execută bloc instrucțiuni
};

În cazul în care sunt implicate mai multe decizii simple care trebuiesc luate funcție de diferite valori ale unei singure variabile, limbajul este prevăzut cu instrucțiune specializată case…. Discuția acestei instrucțiuni se va face ulterior.

Mai jos este prezentat programul corespunzător pseudocodului prezentat anterior.

```
#include <stdio.h>
#include <conio.h>

void main(void)
        {
        float n1,n2,n3,ns,m1,m2;

        clrscr();
        printf("Dati prima nota:");
        scanf("%f",&n1);
        printf("\nDati a doua  nota:");
        scanf("%f",&n2);
        printf("\nDati prima nota:");
        scanf("%f",&n1);
        if( (n1>5) && (n2>5) && (n3>5))
                {
                printf("\nDati  nota de la stat:");
                scanf("%f",&ns);
                m1=(n1+n2+n3)/3;
                m2=(m3+ns)/2;
                printf(" media de licenta %f", m1);
                printf(" media generala  %f",m2);
                }
        else
        printf("Candidat respins");
        getch();
        }
```

Observații:

1. În această fază faptul că s-a folosit o referință, pentru citirea unei variabile (vezi &n1) nu poate fi explicat suficient şi va trebui luat ca atare în cazul citirii variabilelor simple.
2. Operatorul '\n' realizează un salt la linie nouă.

3. Se observă că pe ramura de else nu s-au introdus delimitatorii de bloc (acoladele) pentru că dacă aceştia nu sunt precizați prima instrucțiune este automat considerată aparținând ramurii respective.
4. Pentru programatorii mai avansați: în momentul în care blocul de instrucțiuni este format dintr-o singură instrucțiune se poate folosi operatorul ? ca în exemplul de mai jos, unde :

```
if(a>b) x=25;
else yx=34;
```

poate fi echivalat prin: `a>b (x=25) : (y=34);`

Un alt exemplu de folosire a operatorului ? este o linie de cod care calculează maximul dintre var1 şi var2 şi-l depune în variabila max.

```
max=(var1>var2) ? var1 : var2;
```

Mai jos vom prezenta un program simplu cu exemple despre evaluarea unor expresii compuse într-un program:

```
void main(void)
{
int x = 11,y = 11,z = 11;
char a = 40,b = 40,c = 40;
float r = 12.987,s = 12.987,t = 12.987;
  if (x == y) z = -13;   /* execuția instrucțiunii va avea ca efect z = -13 */
  if (x > z)  a = 'A';   /* execuția instrucțiunii va avea ca efect a = 65  */
  if (!(x > z)) a = 'B'; /* execuția instrucțiunii nu va avea nici un efect */
  if (b <= c) r = 0.0;   /* execuția instrucțiunii va avea ca efect r = 0.0 */
  if (r != s) t = c/2;   /* execuția instrucțiunii va avea ca efect t = 20  */

  if (x = (r != s)) z = 1000; /* execuția instrucțiunii va avea ca efect
                         faptul că x va lua valori pozitive iar z = 1000 */
  if (x = y) z = 222;   /* execuția instrucțiunii va avea ca efect
                    x = y şi z = 222 */
  if (x != 0) z = 333;  /* execuția instrucțiunii va avea ca efect z = 333 */
  if (x) z = 444;       /* execuția instrucțiunii va avea ca efect z = 444 */

  x = y = z = 77;
  if ((x == y) && (x == 77)) z = 33; /* execuția instrucțiunii va avea
                                    ca efect z = 33 */
  if ((x > y) || (z > 12))   z = 22; /* execuția instrucțiunii va avea
                                ca efect z = 22 */
  if (x && y && z) z = 11;          /* execuția instrucțiunii va avea
                                ca efect z = 11 */
 if ((x = 1) && (y = 2) && (z = 3)) r = 12.00; /* execuția instrucțiunii
```

```
                    va avea ca efect     x = 1, y = 2, z = 3, r = 12.00 */
if ((x == 2) && (y = 3) && (z = 4)) r = 14.56; /* execuția instrucțiunii nu
                                        va avea nici un efect */

  if (x == x); z = 27.345;  /* z se va modifica întotdeauna  */
  if (x != x)  z = 27.345;  /* y nu se va modifica nici o dată */
  if (x = 0)   z = 27.345;  /* x va lua valoarea 0 , z va rămâne neschimbat */
}
```

6.2. Instrucțiune de decizie multiplă (selecție)

Există situații în care rezolvarea unei probleme impune ca o decizie să conducă la execuția a mai mult de două ramuri de program. Ca exemplu să considerăm cazul celei mai simple forme de meniu care poate fi oferită unui utilizator.

Se citeşte un număr de la tastatură. Funcție de valoarea dată se poate alege execuția uneia dintre opțiunile prezentate utilizatorului. Mai jos este prezentat un meniu simplu folosind metoda pseudocodului.

```
şterge ecran
afişează mesaj "Apasă una din taste pentru a alege operațiunea dorită"
afişează mesaj "1- Citire din fişier"
afişează mesaj "2- Scriere in fişier"
afişează mesaj "3- Procesare date"
afişează mesaj "4- afişare grafic rezultat"
citeşte o tastă
funcție de tastă citită
        dacă e 1 :  apelează funcția de citire din fişier apoi stop
        dacă e 2 : apelează funcția de scriere in fişier apoi stop
        dacă e 3 : apelează funcția de procesare date apoi stop
        dacă e 4 : apelează funcția de afişare grafică a rezultate apoi stop
        dacă nu e nici unul din numerele de mai sus :
                afişează mesaj "Opțiune invalidă"
```

Se observă că se poate realiza implementarea şi cu instrucțiuni if… dar, acest lucru nu este justificat atât timp cât avem o instrucțiune specializată pentru acest tip de situații. Aceasta este instrucțiunea case… cu următoarea sintaxă:

```
switch(operand) //funcție de operand
    {
    case val1: // dacă operandul ia valoarea 1
                bloc de instrucțiuni 1
            break; // stop
    case val2: // dacă operandul ia valoarea 2
                bloc de instrucțiuni 2
            break; // stop
```

```
                ...
    case valn: // dacă operandul ia valoarea n
                bloc de instrucțiuni n
            break; // stop
    default : // pentru orice altă valoare
            bloc de instrucțiuni ce tratează restul valorilor
            ce le poate avea operandul
    } // stop
```

O formă de reprezentare la nivel de schema logică ar fi cea prezentată în figura 6.2.

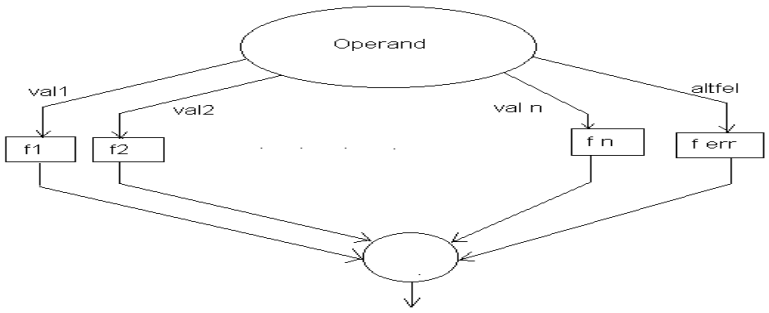

Fig. 6.2. Schema logică pentru reprezentarea unei structuri de selecție multiplă

Mai jos este prezentat un program care citeşte doi operanzi de la tastatură apoi ne permite selecția operației matematice ce trebuie realizată ca în final să afişeze rezultatul.

```
#include <stdio.h>
#include <conio.h>

void main(void)
        {
         int op1, op2;
         float rez;
         char oper;

    clrscr();
        printf("Dati primul operand:");
        scanf("%d",&op1);
        printf("\nDati al doilea operand:");
        scanf("%d",&op2);
        printf("\nSelectionati operatia dorita");
        oper=getch();
        printf("/n1- Adunare");
        printf("/n2- Scadere");
        printf("/n3- Inmultire");
        printf("/n4-Impartire ");
         switch(oper)
```

```
        {
        case ' 1':rez=op1+op2;break;
        case ' 2':rez=op1-op2; break;
        case ' 3':rez=op1*op2; break;
        case ' 4':rez=op1/op2; break;
        default:
                clrscr();
                printf("Operatie nepermisa");
        }
        printf("\nRezultatul este: %d", rez);
}
```