Cursul 5
INSTRUCTIUNILE LIMBAJULUI C

OBIECTIVE

Expresiile sunt utilizate in scrierea instructiunilor. O instructiune este o expresie care se incheie cu simbolul ";". Instructiunile pot fi scrise pe mai multe linii de program, spatiile nesemnificative fiind ignorate. Pe o linie de program putem scrie mai multe instructiuni, simbolul ";" fiind terminator de instructiuni. Instructiunile pot aparea in diferite forme: atribuiri, declaratii, instructiuni conditionale, de ciclare, de salt sau instructiuni compuse.

Instructiunile limbajului C++ sunt:

A) Instructiuni de nivel zero:
- ***vida;***
- ***expresie;***
- ***compusa;***

B) Instructiuni conditionale:
- ***if;***
- ***switch;***
- ***while;***

C) Instructiuni de ciclare:
- ***for;***
- ***do-while;***

D) Instructiunea de transfer al controlului:
- ***exit;***
- ***return;***
- ***continue;***
- ***break;***
- ***goto;***

PREZENTARE GENERALA

INSTRUCTIUNI

INSTRUCTIUNEA *vida*

Se reduce la caracterul "**;**". Nu are nici un efect. Se utilizeaza frecvent in cadrul instructiunilor alternative si repetitive.

INSTRUCTIUNEA *expresie*

Se obtine scriind "**;**" dupa o expresie si are formatul:

```
expresie;
```

Daca expresia din compunerea unei instructiuni expresie este o expresie de atribuire, spunem ca instructiunea respectiva este o *instructiune de atribuire*. Un alt caz frecvent este cel in care expresia este un operand ce reprezinta apelul unei functii. In acest caz instructiunea expresie este o *instructiune de apel* a functiei respective.

Observatie: nu orice expresie urmata de ";" formeaza o *instructiune expresie efectiva*. De exemplu **a;** desi este o instructiune expresie, ea nu are nici un efect.

INSTRUCTIUNEA *compusa*

Este o succesiune de instructiuni incluse intre acolade, succesiune care poate fi precedata si de declaratii:

```
{

     declaratii
     instructiuni

}
```

Declaratiile aici definesc variabile.

INSTRUCTIUNEA *if*

Aceasta instructiune are urmatoarele formate:

forma 1:

```
if(expresie)

     instructiune
```

forma 2:

```
if(expresie)

     instructiune1
     else instructiune2
```

Observatie: modul de utilizare al instructiunii ***if*** din C este identic cu cel - cunoscut voua - din Pascal.

Deoarece o instructiune compusa este considerata ca fiind un caz particular de instructiune, rezulta ca instructiunile din compunerea lui ***if*** pot fi instructiuni compuse. De asemenea instructiunile respective pot fi chiar instructiunea ***if*****.** In acest caz se spune ca instructiunile ***if*** sunt imbricate.

INSTRUCTIUNEA *exit*

Are forma

```
void exit(int cod)
```

isi are prototipul in bibliotecile **stdlib** si **process**. La apelul acestei functii au loc urmatoarele actiuni:

- se videaza zonele tampon ale fisierelor deschise in scriere;
- se inchid toate fisierele deschise;
- se intrerupe executia programului.

Valoarea ‘0’ defineste o terminare normala a programului, iar o valoare diferita de ‘0’ semnaleaza prezenta unei erori.

INSTRUCTIUNEA *while*

Are formatul:

```
while (expresie)

     instructiune
```

Este o instructiune *ciclica conditionata anterior***.** Corpul instructiunii ***while*** este o singura instructiune, care poate fi compusa. Atunci cind sunt compuse spunem ca instructiunile ***while*** sunt imbricate.

INSTRUCTIUNEA *for*

Este asemanatoare instructiunii ***while*** si se utilizeaza pentru a realiza o structura repetitiva conditionata anterior. Formatul ei este:

```
for(exp1;exp2;exp3)
    instructiune
```

INSTRUCTIUNEA *do-while*

Realizeaza structura *ciclica conditionata posterior*. Aceasta instructiune poate fi realizata cu ajutorul instructiunilor definite pana in prezent. Prezenta ei in programe mareste flexibilitatea in programare. Are formatul:

```
do
    instructiune
while(expresie);
```

INSTRUCTIUNEA *continue*

Se poate utiliza numai in cazul unui ciclu. Ea permite abandonarea iteratiei curente. Formatul ei este:

```
continue;
```

Efectul este urmatorul:

- in corpul instructiunii ***do-while*** se abandoneaza iteratia curenta si se trece la evaluarea expresiei care stabileste continuarea sau terminarea ciclului respectiv;
- in corpul instructiunii ***for*** se abandoneaza iteratia curenta si se trece la executia pasului de reinitializare.

Observatie: instructiunea ***continue*** conduce adesea la diminuarea nivelurilor de imbricare ale instructiunilor ***if*** utilizate in corpul ciclurilor.

INSTRUCTIUNEA *break*

Este inrudita cu instructiunea ***continue*** si are formatul:

```
break;
```

Mareste flexibilitatea la scrierea programelor in limbajele C.

INSTRUCTIUNEA *switch*

Permite realizarea structurii *selective.* Aceasta este o generalizare a structurii *alternative*. Ea poate fi realizata prin instructiuni ***if*** imbricate. Structura *selectiva* , in forma in care a fost acceptata, se realizeaza in C cu ajutorul urmatorului format:

```
switch(expresie)

    {

        case c1:

            sir1
            break;

        .........
        case cn:

            sirn
            break;

        default:

            sir

        }
```

Instructiunea ***break*** de la sfirsitul fiecarei alternative, permite ca la intilnirea ei sa treaca la executia urmatoarei instructiuni.

INSTRUCTIUNEA *goto*

Nu este o instructiune absolut necesara la scrierea programelor in C. Cu toate acestea ea se dovedeste utila in diferite cazuri.

Formatul instructiunii:

```
goto nume;
```

-nume- eticheta definita in corpul aceleiasi functii in care se afla instructiunea **goto.**

Prin eticheta intelegem un nume urmat de " **:** " ,dupa eticheta urmeaza o instructiune.

INSTRUCTIUNEA *return*

Admite doua forme:

```
return;
return(exp);
```

Efectul consta in trecerea controlului la functia care a apelat functia respectiva fara transmiterea unei valori in prima varianta sau cu transmiterea unei valori in a doua varianta.