Cursul 4
FUNDAMENTELE LIMBAJULUI C++ (III)

	Obiective	
	Prezentare generala	
Atributele datelor	Clase de memorie	Durata de viata
Accesibilitatea	Operatori	Expresii conditionale

OBIECTIVE

- atributele datelor
- clase de memorie
- durata de viata
- accesibilitate
- operatori si expresii

PREZENTARE GENERALA

ATRIBUTELE DATELOR

Orice data utilizata in cadrul limbajului C, poseda urmatoarele 5 caracteristici:

- **TIPUL** – se determina modul in care datele sunt memorate si operatiile permise cu cu acestea
- **CLASA DE MEMORIE** – specifica locul in care sunt memorate datele
- **DURATA DE VIATA** – reprezinta intervalul de timp in care acestea exista
- **ACCESIBILITATEA** – reprezinta posibilitatea de a accesa date in programe multi-fisier
- **SCOPUL** – reprezinta domeniul de vizibilitate a datelor

CLASE DE MEMORIE<home>

Acest atribut al datelor determina tipul de memorare asociat acestora. Exista 4 categorii de memorare, fiecare fiind specificat utilizand unul din urmatoarele cuvinte-cheie:

auto – date aflate in stiva. In mod implicit, toate variabilele si functiile locale apartin clasei de memorie auto. Astfel, ele apartin stivei asociate functiei apelate (stiva=bloc de memorie de dimansiune finita in care datele sunt tratate dupa principiul "ultimul venit este primul servit") si vor fi apelate in momentul apelului functiei si eliberate la iesirea din functie.

register- date ce pot fi memorate intr-un registru al masinii. Deasemenea, o variabila register este si ea

locala functiei in care a fost definita, dar daca este posibil, va fi depusa intr-un registru al calculatorului.

static – date rezidente in modulul de definitie al acestora. Cuvantul-cheie **static** poate fi, de asemenea, utilizat asociat unor variabile interne oricarei functii. Astfel de variabile nu se regasesc in stiva dar se inscriu in clasa celor statice. Din acest motiv, ele exista pe toata durata executiei programului (asemeni variabilelor globale), dar pot fi accesate numai in interiorul functiilor in care au fost definite.

extern – date definite intr-un modul extern. Aceasta clasa de memorie este rezervata datelor ce se afla in afara oricarei functii.

DURATA DE VIATA

Acest atribut al datelor reprezinta intervalul de timp in care variabila exista, adica cat timp ii este rezervat spatiul in de memorie. Memoria poate proveni din trei surse:

- in cazul variabilelor auto, durata de viata a acestor variabile va fi egala cu durata apelului functiei in cadrul careia au fost definite acestea; variabila va fi depusa in cadrul stivei asociate modulului de definitie sau intr-un registru al calculatorului.
- in cazul variabilelor statice, timpul de viata al acestora este egal cu timpul de executie al intregului program; aceste variabile se vor situa in segmentul de date al programului.
- memoria necesara stocarii unei variabile poate fi alocata dinamic de catre utilizator, caz in care durata de viata a acesteia este controlata de catre programator; variabila va fi depusa in memoria heap (memoria rezervata obiectelor dinamice).

ACCESIBILITATEA<home>

O etapa importanta in stabilirea domeniilor de accesibilitate este legarea atributelor la variabile.Exista doua momente in care se pot executa astfel de legaturi:

- la compilare, caz in care avem legare statica (legare interna)
- in momentul rularii programului, caz in care avem legare dinamica.

In plus, in situatia in care datele apartin unui alt modul si sunt utilizate in cel curent, acestea vor fi externe celui din urma, fapt pentru care, in acest modul, datele vor poseda legare externa. Spre deosebire de clasele de memorie, legarea atributelor actioneaza atat asupra variabilelor, cat si asupra functiilor.

OPERATORI<home>

In marea lor majoritate, actiunile desfasurate in cadrul oricarui program se datoreaza expresiilor formate prin combinatii de date si operatori.

Operatorii limbajului C

<table>
<tr><td>[ ]
&
/
<=
&&
%=
^ =</td><td>( )
*
%
> =
| |
+ =
| =</td><td>.
+
< <
= =
? :
- =
,</td><td>- >
-
> >
! =
=
< < =
#</td><td>+ +
~
<
^
* =
> > =
# #
- > *</td><td>- -
!
>
|
/ =
& =
sizeof</td></tr>
</table>

In functie de numarul de operanzi, operatorii se pot clasifica in trei categorii:

- operatori unari - cu un operand
- operatori binari - cu doi operanzi
- operatori ternari - cu trei operanzi

Precedenta operatorilor in C

<table>
<tr><th>PRIORITATE</th><th>OPERATORI</th><th>EVALUARE</th></tr>
<tr><td>1.</td><td>( ) [ ] -> .</td><td>-></td></tr>
<tr><td>2.</td><td>! ~ + - ++ -- & * (tip) sizeof</td><td><-</td></tr>
<tr><td>3.</td><td>->*</td><td>-></td></tr>
<tr><td>4.</td><td>* / %</td><td>-></td></tr>
<tr><td>5.</td><td>+ -</td><td>-></td></tr>
<tr><td>6.</td><td><< >></td><td>-></td></tr>
<tr><td>7.</td><td>< <= > >=</td><td>-></td></tr>
<tr><td>8.</td><td>== !=</td><td>-></td></tr>
<tr><td>9.</td><td>&</td><td>-></td></tr>
<tr><td>10.</td><td>^</td><td>-></td></tr>
<tr><td>11.</td><td>|</td><td>-></td></tr>
<tr><td>12.</td><td>&&</td><td>-></td></tr>
<tr><td>13.</td><td>| |</td><td>-></td></tr>
<tr><td>14.</td><td>?:</td><td><-</td></tr>
<tr><td>15.</td><td>= *= /= %= += -= &= ^= |= <<= >>=</td><td><-</td></tr>
<tr><td>16.</td><td>,</td><td>-></td></tr>
</table>

EXPRESII CONDITIONALE

Instructiunea :

```
if (a<b) z=a;
else z=b;
```

calculeaza in **z** maximul dintre **a** si **b**. Limbajul C ofera o alternativa de a scrie acest lucru, cu ajutorul operatorului ternar ?: , precum si alte constructii similare.

e1 **?** e2 **:** e3

Expresia **e1** se evalueaza prima; daca ea este adevarata, atunci se evalueaza expresia **e2** si aceasta este valoarea expresiei conditionale; altmiteri, se evalueaza **e3** si aceasta este valoarea expresiei. Numai una dintre expresiile **e2** si **e3** se evalueaza. De exemplu, pentru a pune in **z** maximul dintre **a** si **b**, folosind o expresie conditionala, vom proceda astfel:

```
z= (a<b) ?b:a;
```

Expresiile conditionale conduc adesea la un cod succint. De exemplu, bucla urmatoare tipareste **n** elemente ale unui tablou, 10 pe linie, cu fiecare coloana separata printr-un blanc si cu fiecare linie, inclusiv ultima, terminata cu un singur caracter linie noua.

```
for (i=0;i<n;i)
printf("%6d%c",a[i],(i%10==9||i==n-1)?'\n':' ');
```

Un caracter linie noua se tipareste dupa fiecare al zecelea element si dupa fiecare al **n**-lea element.Toate celelalte elemente sunt urmate de un spatiu. Cu toate ca seamana cu un truc, este instructiv sa incercati sa scrieti lucrul acesta fara a folosi expresia conditionala.