7. PROPRIETĂȚI

În volumul 2, pag. 24, am văzut că **componentele dată** ale unei clase se definesc prin **declarații.** Ele se numesc **date** membru (**member date**) sau **câmpuri** (field).

Proprietățile sunt considerate **intermediare** între **câmpuri** şi **metode**. Ele arată ca şi câmpurile dar acționează ca şi metodele.

Vom accesa o **proprietate** folosind aceeaşi sintaxă ca în cazul accesării **câmpurilor**.

Vom declara o **proprietate** astfel:

```
Modificator_de_acces tip nume_proprietate
{
        ...
}
```

De obicei modificatorul de acces va fi **public**.

Între acolade sunt prezente unul sau două **blocuri accesoare**.

Un **bloc accesor** este un bloc de **citire** sau un bloc de **scriere.**

Blocul accesor de citire conține instrucțiuni care se execută când proprietatea se citeşte, iar **blocul accesor de scriere** conține instrucțiuni care se execută când proprietatea se scrie.

Blocul **accesor de citire** are formatul:

```
get
{
        ...
}
```

El se mai numeşte şi **blocul get** sau **accesorul get**.

Blocul **accesor de scriere** are un format similar:

```
set
{
        ...
}
```

şi el se mai numeşte şi **blocul set** sau **accesorul set**.

Cele trei puncte din blocurile **get** şi **set** reprezintă instrucțiuni.

Din blocul **get** se revine printr-o instrucțiune **return** de forma:

```
return expresie;
```

Valoarea expresiei **expresie** defineşte valoarea curentă a proprietății.

Blocul **set** transferă valoarea proprietății prin intermediul unui **parametru predefinit ascuns** având numele **value (valoare).** Acesta are același tip cu tipul utilizat în declarația de proprietate.

Câmpurile și **proprietățile** se recomandă să aibă un nume care începe cu **literă mare** dacă sunt **publice** și cu **literă mică** dacă au protecția **private**.

Dacă se **utilizează valoarea proprietăți,** de exemplu proprietatea este un operand al unei expresii situată în **partea dreaptă** a operatorului de atribuire, atunci se execută blocul **get** și se revine din el cu o valoare care este valoarea proprietății, adică valoarea operandului respectiv.

În această situație, blocul get joacă rolul unei metode din care se revine cu o valoare a cărei **tip** este **tipul proprietății**.

Dacă proprietatea este în stânga unui operator de atribuire, atunci se va executa blocul **set** ca și cum acesta ar fi o metodă care are ca parametru pe **value**. De obicei, blocul **set** face și testări asupra valorii parametrului **value.**

O declarație de proprietate de obicei conține atât un bloc **get**, cât și un bloc **set**.

Se pot însă defini și proprietăți care **să nu conțină ambele blocuri**. O proprietate a cărei declarație conține doar blocul **get**, se spune că este o proprietate **read-only** (numai – citește). În mod similar, o proprietate a cărei declarație conține doar blocul **set**, se spune că este o proprietate **write-only** (numai – scrie).

Despre proprietățile ale căror declarații conțin ambele blocuri, se spune că sunt proprietăți **read/write** (citire/scriere).

Proprietățile **read/write** sunt cele mai frecvente, iar cele mai rare sunt proprietățile **write-only**.

În final să observăm că **proprietățile** au rolul de a **determina** valoarea unei componente a unui **obiect** (**structură**) prin blocul **get** și/sau de a **seta** valoarea unei componente a unui **obiect** (**structură**) prin blocul **set**.

Blocurile **get** și **set** sunt similare metodelor.

Blocul **get** este similar cu o metodă ce nu are parametri și **returnează** o valoare a cărei **tip** este tipul din declarația proprietății. Blocul **set** este similar cu o metodă care are ca parametru pe **value** de același tip cu tipul din declarația proprietății, metodă care nu returnează nimic (metodă **void**).

Blocurile **get** și **set** pot și ele să aibă un **modificator de acces** propriu care să difere de modificatorul de acces de la începutul declarației de proprietate:

```
modificator_de_access_get get {...}
modificator_de_access_set set {...}
```

De exemplu, dacă modificatorul de acces al declarației de proprietate este **public**, atunci modificatorii de acces ale blocurilor **get** și /sau **set** pot fi **private** dar nu și invers.

Evident, dacă blocurile **get** şi/sau **set** nu au precizat un modificator de protecție, atunci ele au în mod automat acelaşi modificator de protecție ca şi al declarației de proprietate.

Exerciții

7.1. Să se scrie o aplicație care citeşte trei numere oarecare pozitive ce reprezintă măsurile laturilor unui triunghi şi afişează aria triunghiului respectiv.

Proiectul aplicației va avea numele **Proprietati1.**

Aria unui triunghi oarecare se poate calcula folosind formula lui **Heron**. Dacă **a, b** şi **c** sunt măsurile laturilor unui triunghi şi notăm cu **p** semiperimetrul triunghiului (p = (a+b+c)/2), atunci aria triunghiului respectiv se calculează după relația:

$$aria = \sqrt{p(p-\ )(p-\ )(p-\ )}$$

Din geometrie se ştie că pentru ca **a, b** şi **c** să reprezinte măsurile laturilor unui triunghi, este necesar şi suficient ca **p-a > 0, p-b > 0** şi **p-c > 0.**

Aplicația are două clase: clasa cu numele **Program** şi clasa cu numele **triunghi.**

Clasa **triunghi** are ca date membru variabilele **a, b, c** şi **p** care sunt câmpuri ale clasei respective. Clasa triunghi are şi un constructor cu **parametri a, b,** şi **c** pentru a **iniția liza câmpurile cu acelaşi nume** la instanțierea unui obiect al clasei respective.

De asemenea, clasa **triunghi** are proprietățile **A, B, C, P,** şi **Aria**.

Proprietățile **A, B** şi **C** au numai blocul **get** care realizează accesul la câmpurile **a, b** şi **c** ale sale.

Proprietatea **P** are numai blocul **set** prin care se testează condițiile ca semiperimetrul triunghiului ((a+b+c)/2) să fie mai mare decât măsurile laturilor sale. În caz afirmativ, semiperimetrul triunghiului se atribuie lui **p**. Valoarea semiperimetrului este dată de parametrul **value**. Dacă condiția de mai sus nu este satisfăcută, se afişează un mesaj de eroare corespunzător şi lui **p** i se atribuie valoarea **zero**.

Proprietatea **Aria** are numai blocul **get**. Blocul respectiv calculează şi returnează valoarea ariei triunghiului.

Clasa **Program** are două metode membru statice: **cit** şi **Main**.

Metoda **cit** citeşte de la tastatură un număr de tip **double** într-un bloc **try**.

Se semnalează eroare dacă numărul citit nu este pozitiv sau dacă nu s-a tastat un număr oarecare.

V-239

Limbajul C# pentru începători
Mediul de Programare VISUAL STUDIO # 2008
depanarea programelor şi tratarea erorilor
proprietăți, tablouri şi clasa Array

Liviu Negrescu, Lavinia Negrescu

1. MEDIUL DE PROGRAMARE *VISUAL STUDIO C# 2008*
Rezumat

2. TIPURI DE DATE
2.1. Tipuri structurate
2.1.1. Tipuri structurate corespunzătoare tipurilor primitive
2.2. Tipul enumerare
2.3. Tipul referință
2.4. Tipul pointer
Rezumat

3. DEPANAREA PROGRAMELOR
3.1. Erori sintactice
3.2. Erori logice
3.2.1. Modul de execuție în regim de depanare
Rezumat

4. EXCEPȚIILE ŞI TRATAREA ERORILOR
4.1. Construcțile *try*, *catch* şi *finally*
4.1.1. Depăşiri la evaluarea expresiilor
4.2. Instrucțiunea *throw*
Rezumat

5. DATE CONSTANTE
Rezumat

6. CONVERSIA DATELOR FOLOSIND METODA *PARSE*
Rezumat

7. PROPRIETĂȚI
Rezumat

8. OPERATORII *IS*, *AS* ŞI *TYPEOF*
8.1. Operatorul *is*
8.2. Operatorul *as*
8.3. Operatorul *typeof*

Rezumat

9. UTILIZAREA SPAȚIILOR DE NUME
Rezumat

10. TABLOURI
10.1. Tablouri de obiecte
10.2. Copierea tablourilor
10.3. Clasa *Array*
Rezumat