**Упражнение 1- дървета, корен, възли-листа, междинни възли**

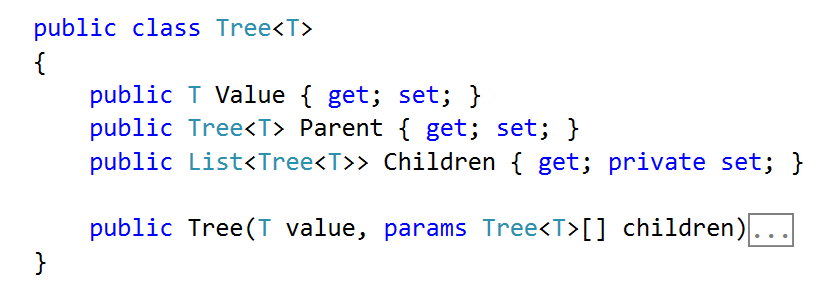
**Зад 1.** **Намиране на корен на дърво**

*Напишете програма, която прочита дърво от N възела, представено като набор от N-1 двойки възли (възел-родител, възел дете) и намира корена му:*

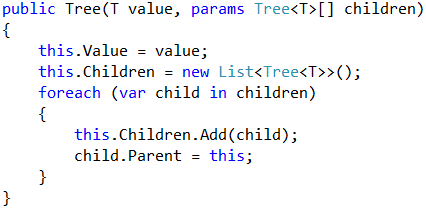
|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Дърво** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Корен: 7 |  |

### Подсказки

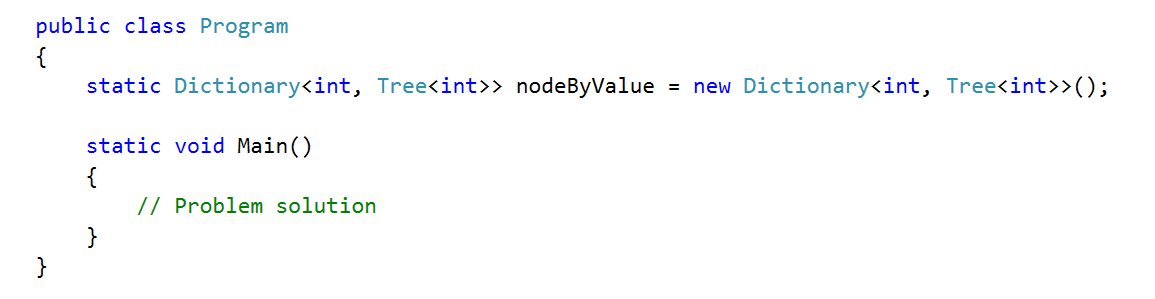
Използвайте рекурсивна дефиниция **Tree<T>**. Пазете **стойност**, **родител** и **деца** за всеки възел от дървото:



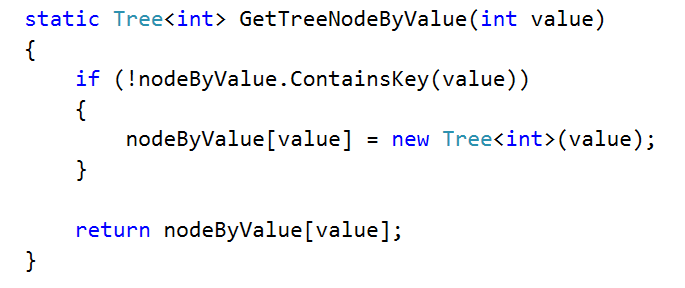
Променете конструктора на **Tree<T>** така че да може да се присвои родител за всеки възел-дете:



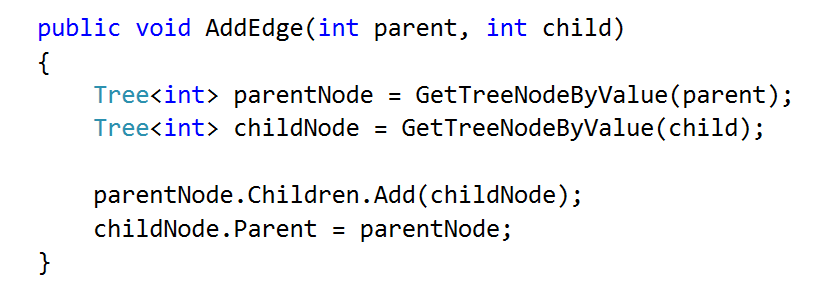
Използвайте **речник**, за да пазите колекция от възлите и техните стойности. Това ще ви позволи да намирате много по-бързо възлите на дървото по време на неговото конструиране:



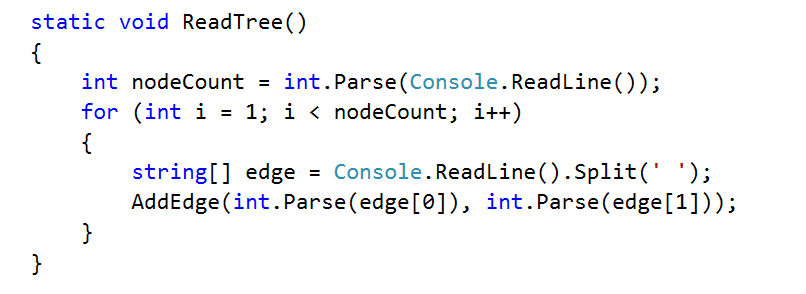
Напишете метод за намиране на възел от дървото по неговата стойност или ако не съществува да създава нов възел:



Създайте метод за добавяне на връзка в дървото.



Създайте дървото. Дадени са ви връзките между възлите в дървото (родител + дете). Използвайте речника за да намирате децата и родителите по техните стойности:



Накрая можете да намерите корена - той няма родител.

## Зад 2. Възли – листа

*Напишете програма, която намира всички възли-листа и ги отпечатва на стандартния изход.*

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Дърво** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Листа: 1 6 12 21 23 31 |  |

### Подсказки

*Намерете всички възли, които нямат деца.*

## Зад 3. Междинни възли

*Напишете програма, която прочита дървото и намира всички междинни възли (в нарастващ ред).*

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Дърво** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Междинни възли: 14 19 |  |