Logica de bază care se foloseşte atunci când se scrie un program nu depinde de limbajul de programare, tehnologia sau setul de instrumente care se utilizează pentru dezvoltare. Cu alte cuvinte, este universal modul în care se formulează logica de programare. Prin urmare, cunoașterea unui limbaj de programare se reduce în primul rând la particularitățile care ţin de sintaxă, adică la ceea ce diferențiază limbajele. Aşadar, în lecțiile din cadrul acestui modul se va acorda atenție abordărilor logice universale care se utilizează atunci când se scriu programele. Înțelegerea unor astfel de abordări va fi foarte utilă pentru a obține o imagine mai amplă a structurii programelor și a modului în care acestea funcționează.

În această lecție va fi prezentat lucrul cu valorile, adică cu datele. Astfel, veți avea ocazia să învăţaţi ce reprezintă variabilele, tipurile de date și operatorii.

Stocarea datelor în memorie

Miezul fiecărui limbaj de programare este capacitatea procesării a datelor. Aşadar, datele care se prelucrează pot fi de diferite tipuri: numere, text, caractere etc. Se poate spune că programele, în timpul funcționării, folosesc diferite date. Programele de afaceri pot gestiona cu datele angajaților, clienților, produselor și toate acele concepte care sunt importante pentru afacere. Aplicația meteo gestionează cu datele privind temperatura, umiditatea, latitudinea și longitudinea și cu datele meteo. Jocurile PC folosesc sprite-uri sau poligoane care alcătuiesc grafica pe care utilizatorul o vede pe ecranul său, pe display.

Indiferent de tipul de date cu care gestionează un program, este necesară stocarea acestor date în memoria calculatorului. Memoria calculatorului poate fi experimentată ca un spațiu în care pot fi stocate valorile. Fiecare program în timpul execuției sale folosește așa-numita memorie de lucru pentru a stoca datele.

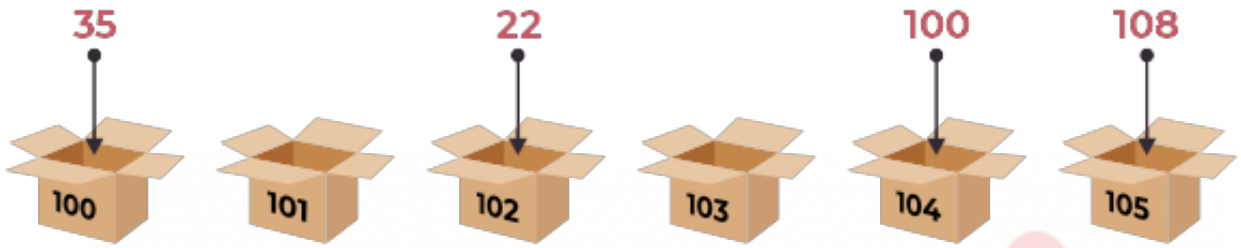
Memoria de lucru poate fi experimentată precum o multitudine de cutii (imaginea 4.1.).

Imaginea 4.1. Locaţii de memorie

Fiecare cutie reprezintă o locație de memorie în care este posibilă înscrierea unor date. Fiecare locație de memorie dispune de propria adresă (imaginea 4.2.).

Imaginea 4.2. Locaţii de memorie şi adresele lor

Cifrele pe care le puteți vedea în imaginea 4.2. (100, 101, 102, 103 etc.) reprezintă adrese de memorie. La fiecare adresă de memorie este posibil să înscrieți o oarecare valoare (imaginea 4.3.).



Imaginea 4.3. Plasarea datelor la anumite locaţii de memorie

Având în vedere că execuția oricărui program este însoțită de citirea și înscrierea constantă a valorilor, nu este greu de concluzionat că manipularea cu astfel de valori este condiționată de cunoașterea locației de memorie în care este înscrisă o astfel de valoare. În timpul scrierii programului, astfel de locații de memorie pot fi accesate în mod indirect, prin intermediul variabilelor.

Ce reprezintă variabilele?

Variabilele reprezintă simboluri prin care se marchează locațiile de memorie. Datorită lor, nu este necesar să reţinem adresele de memorie unde se află datele. După ce o valoare este stocată în memorie, aceasta este reprezentată printr-o variabilă.

În limbajul de programare Python, o valoare poate fi prezentată în modul următor:

```
x=35
```

În limbajul de programare Java, același lucru se poate atinge după cum urmează:

```
int x=35;
```

În ambele cazuri este creată variabila cu denumirea x și valoarea 35. Aceasta înseamnă practic că valoarea 35 în restul programului va fi reprezentată de variabila cu denumirea x. Prin urmare, la ieşire, o astfel de valoare este posibil să o imprimați în modul următor (Python):

```
x=35
print(x)
```

În Java, în mod identic, poate fi atins după cum urmează:

```
int x=35;
System.out.print(x);
```

Procesul de creare a unei variabile și plasarea unei valori în ea reprezintă o acțiune care constă din două părți care se numesc: declarare și inițializare

Declararea variabilei poate fi văzută precum rezervarea unui spațiu de memorie în care ulterior va fi plasată valoarea (imaginea 4.4.).

Imaginea 4.4. Procesul de declarare – rezervarea spaţiului pentru stocarea valorii

Inițializarea poate fi văzută ca plasarea valorii într-un spațiu de memorie de rezervat (imaginea 4.5.).

Imaginea 4.5. Procesul de iniţializare – atribuirea unei valori variabilei

În ambele exemple anterioare de creare a variabilei x, declararea și inițializarea au fost întreprinse laolaltă. În limbajul Python, acest lucru este comun, în timp ce în limbajul Java, declararea și inițializarea se întreprind în mod separat:

```
int x;
x = 35;
```

Prima linie ilustrează declararea, în timp ce cea de-a doua reprezintă inițializarea.

Puteți încerca codul pe care l-aţi văzut şi care se referă la crearea variabilelor și imprimarea valorilor acestora în mediul de execuție a codului. Iată primul astfel de mediu pentru limbajul Python:

```
x=35
print(x)
```

Încercați să schimbați în mediu valoarea 35 cu o altă valoare numerică și veți avea ocazia să vedeţi la ieșire o imprimare diferită.

Mediul de execuție a codului Java este următorul:

```
public class JavaHelloWorld
{

 public static void main(String[] args)
 {

  int x=35;
  System.out.print(x);
```

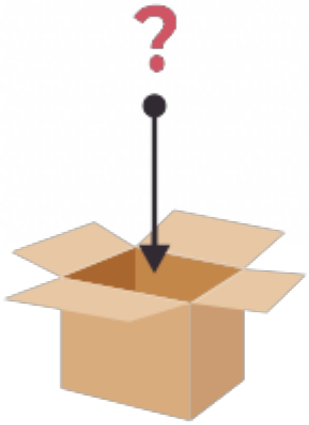
```
 }

}
```

Codul prezentat îl puteți încerca în compilatoarele afișate. Ţineţi cont de faptul că, în limbajul Java, codul care va fi afișat este necesar să îl scrieți în loc de linia int `x=35;` ?i `System.out.print(x);`

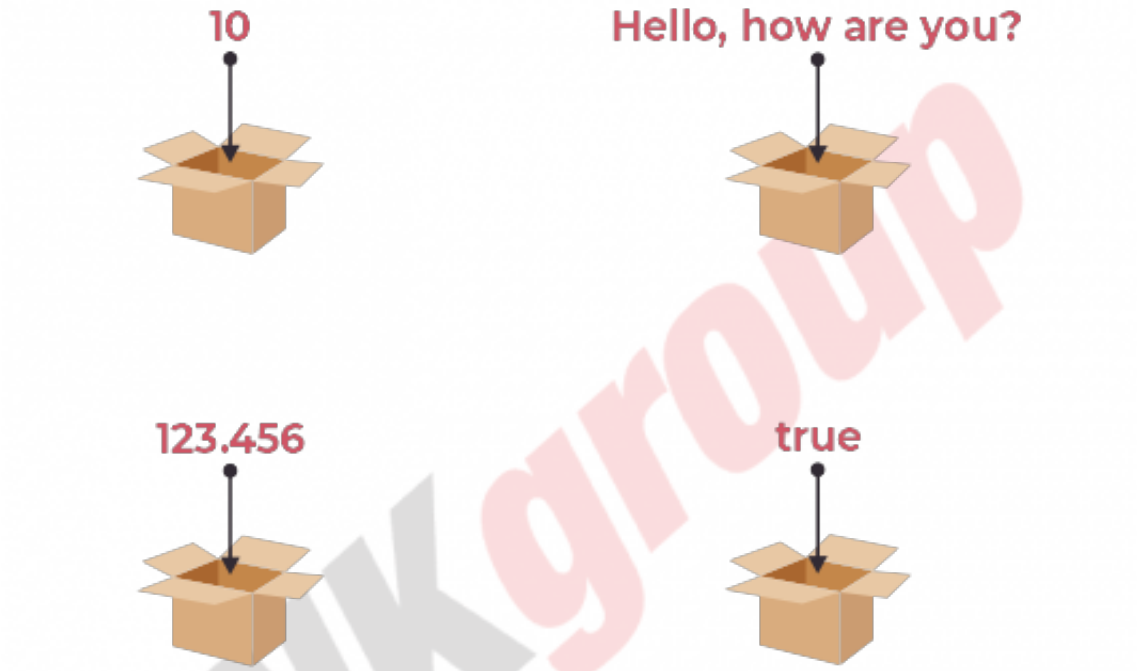
Ce reprezintă tipurile de date?

Atunci când se lucrează cu valori într-un singur program, adică când se manipulează cu astfel de valori datorită variabilelor, una dintre cele mai importante întrebări este ce se poate plasa în astfel de variabile (imaginea 4.6.).



Imaginea 4.6. Ce valori pot fi plasate într-o locație de memorie?

Clasificarea valorilor care pot fi plasate în variabile se efectuează datorită tipurilor de date (imaginea 4.7.).



Imaginea 4.7. Tipuri de date

Tipurile de date permit ca în program să fie reprezentate diferite tipuri de valori - caractere, text, diverse tipuri de numere, valori logice etc. În imaginea 4.7. puteți vedea patru valori diferite care sunt plasate într-o cutie – număr întreg , text, număr zecimal și valoare logică. Ceea ce imaginea 4.7. ilustrează în limbajul Python este posibil să se atingă în modul următor:

```
a = 10
b = "Hello, how are you?"
c = 123.456
d = true
```

În limbajul de programare Java, definirea valorilor necesită și evidențierea explicită a tipurilor acestora:

```
int a = 10;
string b = "Hello, how are you?";
double c = 123.456;
boolean d = true;
```

Pe baza codului prezentat, nu este greu de concluzionat că, atunci când se lucrează cu tipurile de date, în funcție de limbaj, există diferențe semnificative. Diferențele pe care tocmai le-ați văzut provin din diferitele sisteme de manipulare cu tipurile care există în limbajele Java și Python. Două feluri de sisteme de tip sunt următoarele:

- **sistem de tip dinamic** – verificarea tipului se efectuează în timpul execuției programului; tipurile se determină pe baza unei valori definite și aceasta este întreprinsă numai de mediul de execuție; programatorul nu are obligația să definească tipurile, iar un exemplu clasic de limbaj cu un sistem de tip dinamic este Python, iar pe lângă acesta, un astfel de sistem de tip au şi limbajele JavaScript, Lisp, PHP, Ruby, Perl, Lua etc.;
- **sistem de tip static** – tipurile se verifică în timpul translației și fiecare variabilă trebuie să dispună de propriul tip, astfel că este responsabilitatea programatorului să definească tipul atunci când declară o variabilă; sistemul de tip static este una dintre caracteristicile de bază ale limbajului Java, dar și al numeroaselor altor limbaje - C, C++, C#, Kotlin, Fortran, Go, Pascal, Swift etc.

Tipuri de date de bază

Este important de înțeles că, chiar și atunci când sistemul de tipuri este dinamic, nu înseamnă că tipurile nu există, ci doar că programatorul nu

manipulează cu ele în mod direct. Astfel, ajungem la concluzia că toate limbajele de programare cunosc conceptul de tipuri, iar un anumit grup de tipuri de bază este comun pentru toate limbajele. Se poate spune că tipurile de bază care există în fiecare limbaj sunt următoarele:

- numere,
- text și caractere,
- valori logice.

Numere

Toate limbajele de programare permit prezentarea numerelor odată cu scrierea codului sursă. Numerele pot fi definite în diferite forme, precum numere întregi, numere cu zecimale, numere binare și hexazecimale etc.

Un exemplu de definire a mai multor variabile, ale căror valori sunt numere în limbajul Python, ar putea arăta astfel:

```
intVal = 26
floatVal = 26.23
octVal = 0032
hexVal = 0x1a
binVal = 0b11010
```

În limbajul de programare Java, același lucru se poate efectua după cum urmează:

```
int decVal = 26;
double floatVal = 26.23;
int octVal = 032;
int hexVal = 0x1a;
int binVal = 0b11010;
```

Text și caractere

Pe lângă numere, datele sub formă de text sunt deosebit de importante. În Python, textul poate fi definit după cum urmează:

```
someTxt = "Hello World"
```

În limbajul Java, acelaşi lucru poate fi realizat în modul următor:

```
String someTxt = "Hello World";
```

Puteți vedea că caracteristica de bază a valorilor de tip text este existența ghilimelelor între care se definesc astfel de valori. Astfel de ghilimele sunt lucrul datorită căruia se disting datele de tip text de datele numerice.

Tipul de date de tip text, în aproape toate limbajele, se numeşte **string**.

Unele limbaje cunosc și termenul caracter atunci când vine vorba de tipurile de date. Astfel este și limbajul Java, în care un caracter poate fi definit după cum urmează:

```
char someChar = 'X';
```

În limbajul de programare Java, caracterele se definesc între ghilimele simple, adică apostrof. Limbajul de programare Python nu recunoaște termenul caracter, în acest limbaj caracterele sunt reprezentate de tipul string, a cărui lungime este de un caracter.

Valori logice

Baza fiecărui limbaj de programare este tipul de date logic, care permite definirea datelor care pot avea două forme: true/false, on/off, 0/1 etc. Este vorba despre un tip care, în majoritatea limbajelor de programare, se numește **boolean** sau **bool**.

Tipul de date logic dispune de doar două valori - `true` şi `false`, atunci când vorbim despre limbajul Java, și valorile `True` şi `False`, atunci când vorbim despre limbajul Python.

Declararea unei variabile de tip boolean în limbajul Python se poate efectua în modul următor:

```
x=True
```

În limbajul de programare Java, este necesar să se definească tipul:

```
boolean x = true;
```

Valorile logice se pot obține și prin comparație simplă (Python):

```
print(9>10)
```

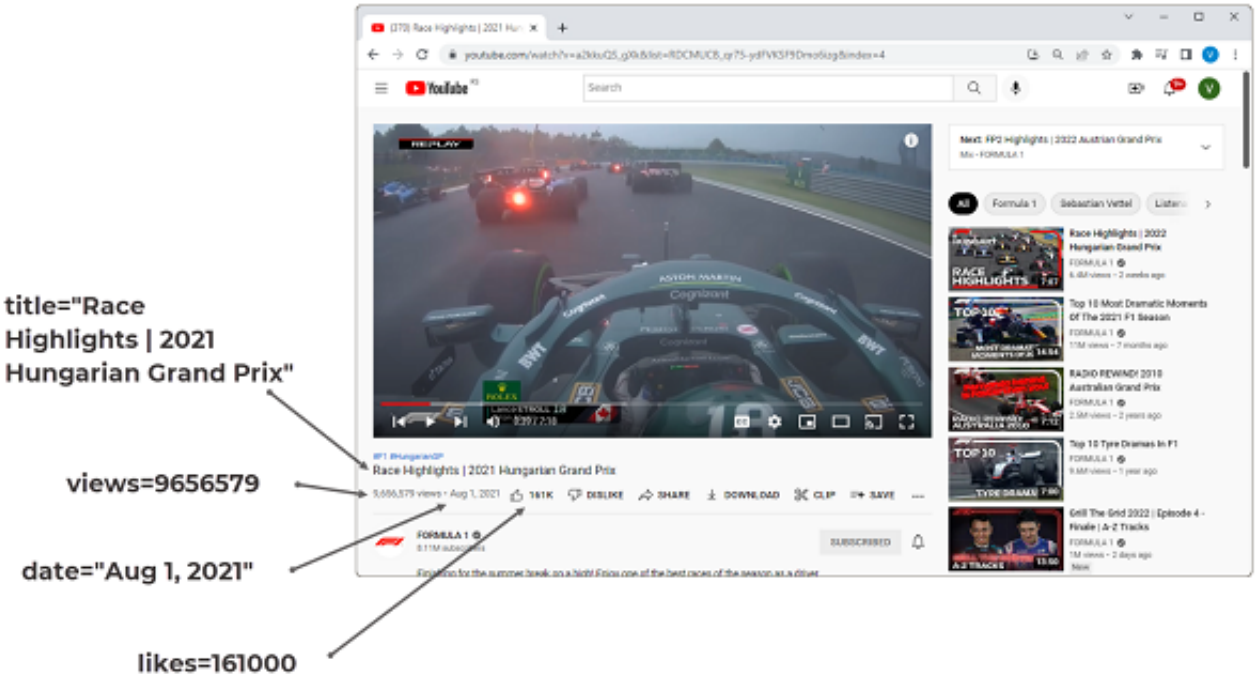
Linia de cod afișată va avea ca efect imprimarea valorii `False`, deoarece numărul 9 nu este mai mare decât numărul 10.

Tipurile de date logice sunt deosebit de importante pentru controlul fluxului de execuție a programului, care este un alt subiect foarte important în programare, indiferent de limbajul care se foloseşte. Lecția următoare va fi dedicată în întregime acestui subiect, atunci când vom întâlni din nou tipurile de date logice.

De ce avem nevoie de variabile?

În rândurile anterioare au fost prezentate cele mai importante proprietăți ale variabilelor și tipurilor de date în programare. Este posibil ca, în momentul de faţă, să nu puteți vedea importanța variabilelor în dezvoltarea software-ului. Din acest motiv, vom prezenta câteva exemple reale privind utilizarea variabilelor în programe (aplicații) populare care se utilizează zilnic.

Primul exemplu ilustrează platforma populară de partajare a videoclipurilor Youtube (imaginea 4.8.).



Imaginea 4.8. Diferite variabile într-un exemplu al unui videoclip Youtube

În imaginea 4.8. puteți vedea pagina pentru vizionarea videoclipului pe platforma YouTube. În imaginea 4.8. sunt marcate doar câteva variabile - titlu (`title`), vizualizări (`views`), data publicării (`date`) și numărul de like-uri (`likes`). Practic, toate datele pe care le puteți

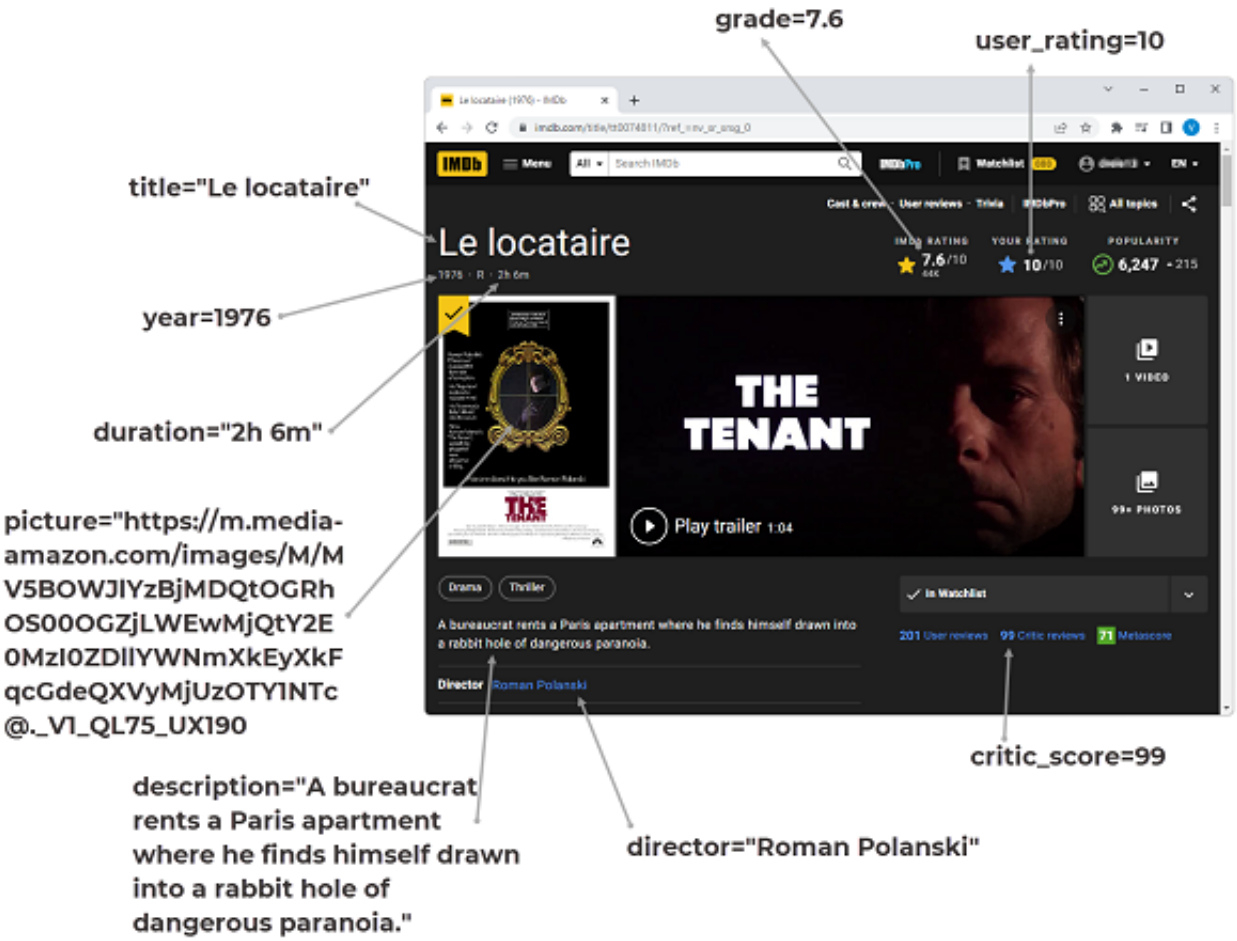
vedea pe o pagină în codul sursă sunt prezentate prin intermediul variabilelor.

Următorul exemplu va ilustra utilizarea variabilelor pe exemplul unui joc video (imaginea 4.9.).

Imaginea 4.9. Variabile în exemplul unui joc video

În imaginea 4.9. puteți vedea o afișare a unui joc video. Suma totală de bani pe care a câștigat-o jucătorul (`balance`), energia (`health`), scutul (`armour`) și numărul de gloanțe (`bullets`) sunt exemple clasice de valori reprezentate prin variabile. De data aceasta toate variabilele sunt de tip numeric.

În final, iată un alt exemplu de variabile, de data aceasta ilustrat pe un site web popular (imaginea 4.10.).



Imaginea 4.10. Variabile în exemplul afişării unui film pe site-ul web Imdb.com

În imaginea 4.10. puteți vedea numeroase variabile prin care sunt prezentate informațiile despre un site web.

Exerciţii

```
public class JavaProgram
{

 public static void main(String[] args)
```

```
{
 //todo
}
}
```

Exerciţiul 1

Într-un program Java, este necesar să se definească valori prin care se vor prezenta anii și numele și prenumele unui angajat. Numărul care ţine de ani este 47, iar numele şi prenumele sunt John Lord.

Exerciţiul 2

Valorile din exercițiul anterior acum trebuie prezentate într-un program Python.

Exerciţiul 3

Valorile prezentate în exerciţiile anterioare trebuie imprimate la ieşire.

Notă

Dacă întâmpinați unele greutăţi când rezolvaţi exercițiile sau doriți să verificați codul pe care l-ați scris, puteți descărca soluțiile exercițiilor

de la următorul link.

