Obiectivul de bază al oricărui program este de a modifica într-un mod oarecare datele care se utilizează în timpul execuției programului. Astfel de procedură se mai numește procesarea datelor, iar mecanismele de bază ale procesării datelor se bazează pe utilizarea operatorilor. Aşadar, în această lecție ne vom familiariza cu elementele de limbaj care permit efectuarea diverselor operații asupra valorilor. Este vorba despre operatori.

Ce reprezintă operatorii?

Operatorii reprezintă elemente lexicale speciale ale limbajului, care permit efectuarea operațiilor asupra valorilor. Operatorii se pot diviza în câteva grupuri:

```
x = 10
y = 5

print(x + y)
print(x - y)
print(x * y)
print(x / y)
```

- operatori aritmetici,
- operatori de atribuire,
- operatori de comparare,
- operatori logici.

În funcție de numărul de operanzi, operatorii se pot diviza în următoarele grupuri:

- operatori unari – operează asupra unui singur operand,
- operatori binari – operează asupra doi operanzi,
- operatori ternari – necesită trei operanzi.

În continuare vă vom prezenta utilizarea celor mai importanți operatori

care se pot utiliza atunci când se scrie un program.

Operatori aritmetici

Operatorii aritmetici sunt cei care efectuează operații aritmetice asupra valorilor. Tabelul 5.1. prezintă cei mai importanţi operatori aritmetici.

Operator	Semnificaţie
+	adunare
-	scădere
*	înmulţire
/	împărţire
%	modul (restul împărţirii)

Tabelul 5.1. Operatori aritmetici

Cei patru operatori ai operațiilor aritmetice standard, la fel ca în matematică, efectuează adunarea, scăderea, înmulțirea și împărțirea. Exemplul privind utilizarea lor poate fi următorul (Python):

```
x = 10
y = 5

print(x + y)
print(x - y)
print(x * y)
print(x / y)
```

În limbajul de programare Java, ceva similar se poate întreprinde după cum urmează:

```
public class JavaHelloWorld
{

 public static void main(String[] args)
 {

int x = 10;
int y = 5;

System.out.println(x + y);
System.out.println(x - y);
System.out.println(x * y);
System.out.println(x / y);

 }

}
```

În exemplele prezentate, mai întâi au fost create două variabile întregi cu valorile 10 și 5. Apoi asupra lor s-au utilizat operatorii aritmetici pentru adunare, scădere, înmulțire și împărțire. Codul produce următoarea ieșire:

```
15
5
50
2
```

Atunci când se întreprind operații aritmetice, trebuie ţinut cont de ordinea executării lor. Operațiile aritmetice au aceeași prioritate de execuție ca în matematică. Înmulțirea și împărțirea au prioritate față de adunare și scădere. Ordinea de execuție a operațiilor poate fi modificată atunci când se folosesc paranteze.

Operatori de atribuire

Operatorii de atribuire atribuie valori unei variabile. Operatorul de atribuire de bază este caracterul egal (=). Pe lângă acesta, există şi alți operatori de acest tip, care reprezintă o combinație de operatori de atribuire și operatori aritmetici. Cei mai importanți operatori de atribuire sunt prezentați în tabelul 5.2.

Operator	Descriere	Exemplu	Semnificaţie
=	atribuie valoarea din dreapta variabilei din stânga	x = y	x = y
+=	majorează valoarea variabilei x cu valoarea y și atribuie valoarea nou obținută variabilei x	x += y	x = x + y
-=	micşorează valoarea variabilei x cu valoarea y și atribuie valoarea nou obținută variabilei x	x -= y	x = x - y
*=	înmulțește valoarea variabilei x cu valoarea y și atribuie valoarea nou obținută variabilei x	x *= y	x = x * y
/=	divizează	x /= y	x = x / y

valoarea variabilei x cu valoarea y și atribuie valoarea nou obținută variabilei x

Tabelul 5.2. Operatori de atribuire

După cum s-a menționat deja, operatorul de atribuire de bază este caracterul egal (=) și ne-am familiarizat cu utilizarea lui de mai multe ori (Python):

```
a = 10
myString = "Hello World"
z = 3.33
```

Acesta este un exemplu de atribuire a valorilor diferitelor variabile. Efectul identic în Java poate fi obținut după cum urmează:

```
int a = 10;
String myString = "Hello World";
int z = 3.33;
```

În tabelul 5.2. puteți vedea că operatorul de atribuire standard poate fi combinat cu operatorii aritmetici şi astfel se pot produce efecte interesante. Iată un exemplu de combinare a operatorului de adăugare și a operatorului de atribuire (Python):

```
x = 10
x += 5
print(x)
```

În Java, ceva similar se poate realiza după cum urmează:

```
int x = 10;
x += 5;
System.out.println(x);
```

În exemplu, mai întâi s-a întreprins declararea și inițializarea valorii x, cu valoarea 10. În cea de-a doua comandă, valoarea variabilei x a fost majorată cu 5, obținându-se astfel o nouă valoare atribuită variabilei x. În cele din urmă, codul de ieșire produce următorul efect:

15

Efectul tocmai obținut ar putea fi obținut și în modul următor (Python):

```
x = 10
x = x + 5
print(x)
```

Echivalentul în Java este următorul:

```
int x = 10;
x = x + 5;
System.out.println(x);
```

Diferența față de exemplul anterior constă în cea de-a doua comandă, unde acum este folosită sintaxa care presupune un pas în plus. Cu alte cuvinte, acum puteți vedea clar divizarea tuturor pașilor care au fost ascunși în exemplul anterior utilizând operatorii unificați de adăugare și atribuire. Respectând acelaşi principiu funcționează şi ceilalți

operatori, care reprezintă o combinație între o operație aritmetică și una de atribuire.

Operatori de comparație

Operatorii de comparație se utilizează pentru a compara două sau mai multe valori. Prin comparație se creează o nouă valoare care indică asupra rezultatului de comparație. O astfel de valoare este de tip logic (`boolean`).

Operatorii de comparație se utilizează în declarațiile logice pentru a determina adevărul unei expresii. Tabelul 5.3. prezintă cei mai importanți operatori de comparație.

Operator	Semnificaţie
==	este egal
!=	nu este egal
<	mai mic decât
>	mai mare decât
>=	mai mare decât sau egal cu
<=	mai mic decât sau egal cu

Tabelul 5.3. Operatori de comparație

Exemplul de bază privind utilizarea operatorului de comparație ar putea arăta astfel în Python:

```
value1 = 1
value2 = 2
print(value1 == value2)
```

Echivalentul Java este următorul:

```
int value1 = 1;
int value2 = 2;
System.out.println(value1 == value2);
```

În exemplu sunt create două variabile întregi cu valorile 1 și 2. Apoi se verifică egalitatea lor folosind operatorul de comparație care determină egalitatea, iar rezultatul se imprimă la ieşire:

```
False
```

Valorile 1 și 2 nu sunt egale și, aşadar, ca rezultat al acestei comparații se obţine valoarea `false`.

Ceilalţi operatori de comparație se pot utiliza conform aceluiași principiu:

```
print(value1 != value2)
```

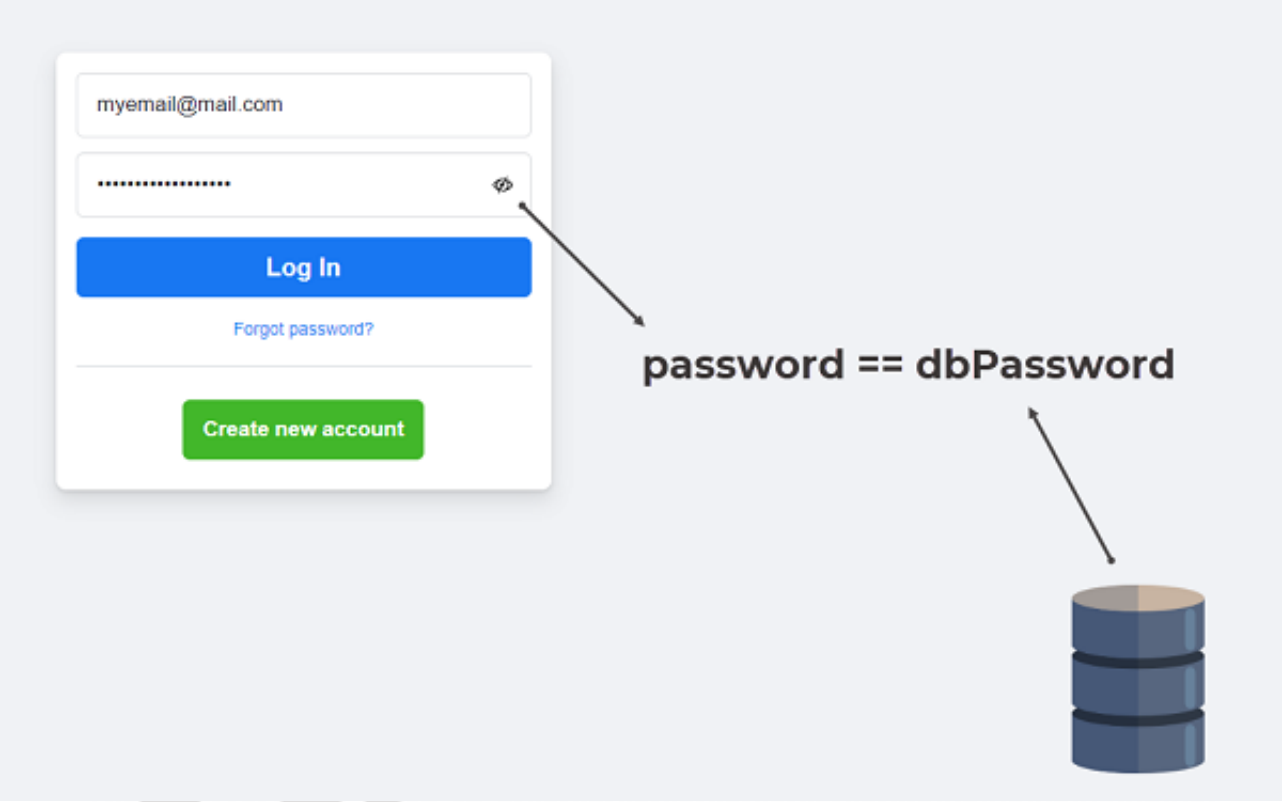
Acesta este acum un exemplu în care se imprimă inegalitatea, aşadar, la ieşire se obţine rezultatul:

```
True
```

Valorile 1 și 2 nu sunt egale, de unde provine valoarea `True`. Expresia ar putea fi citită astfel: *Sunt valorile variabilelor value1 și value2 diferite?* Răspunsul este, desigur, da, afirmativ, de unde provine și valoarea `True` ca rezultat.

Exemple privind utilizarea operatorilor de egalitate și inegalitate

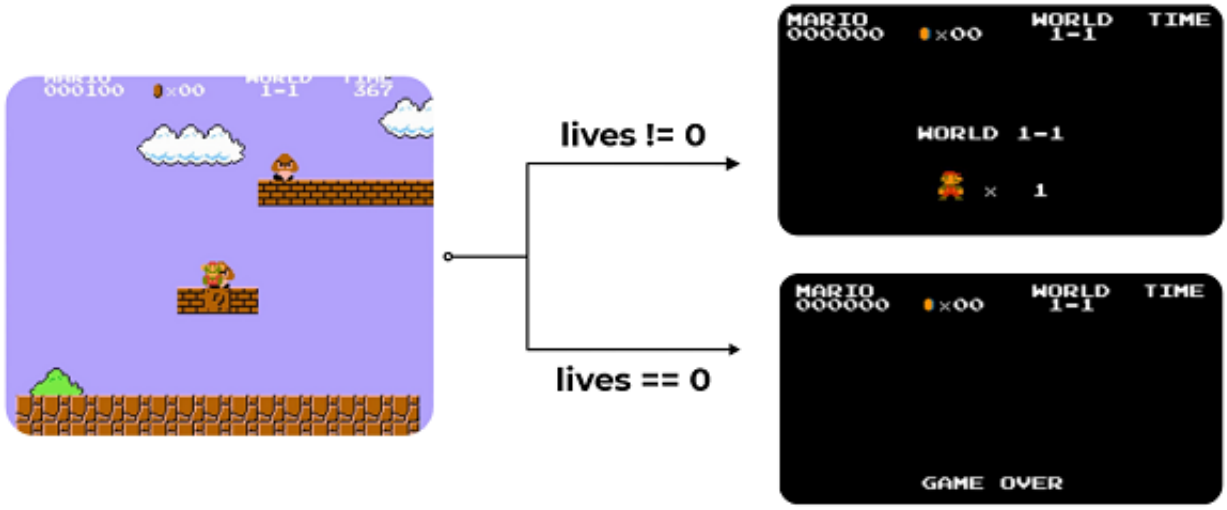
În acest moment, poate vă este dificil să înțelegeți importanța pe care o au operatorii atunci când se scrie un program. Prin urmare, prezentarea diferiților operatori va fi însoţită de exemple reale. Pentru început, exemple privind operatorii de egalitate și inegalitate.



Imaginea 5.1. Exemplu privind utilizarea operatorilor de egalitate

În imaginea 5.1 puteți vedea o situație zilnică care se întâlneşte pe web. Este vorba despre un formular de autentificare în care utilizatorul își introduce datele de acces. Parola (*password*) introdusă de utilizator se compară cu cea introdusă în baza de date. O astfel de comparație se poate efectua tocmai cu operatorul de egalitate. În cazul unei potriviri, va produce valoarea `True` și invers.

Ca următorul exemplu de utilizare a operatorilor de egalitate și inegalitate poate servi situația din jocul video Super Mario (imaginea 5.2.).



Imaginea 5.2. Exemplu privind utilizarea operatorilor de egalitate

La începutul jocului, Super Mario are mai multe vieți. Când atinge un inamic, Super Mario pierde o viață. Ceea ce se întâmplă după aceea, este direct condiţionat de numărul de vieți disponibile pentru protagonist. Dacă numărul de vieți este diferit de zero, jucătorul primește ecranul de mai sus. Dacă numărul de vieți este 0, jucătorului îi este prezentat mesajul Game Over. În ambele scenarii participă operatorii de comparație, în primul operatorul de inegalitate, iar în cel de-al doilea operatorul de egalitate.

Utilizarea operatorului < și > este cunoscută din matematică:

```
print(value1 < value2)
print(value1 > value2)
```

Acest cod produce următorul efect:

```
True
False
```

Valoarea variabilei `value1` este mai mică decât valoarea variabilei `value2`, aşadar, prima valoare este `True`, iar cea de-a doua este `False`.

Exemple privind utilizarea operatorilor mai mare decât și mai mic decât

Utilizarea operatorilor mai mare decât și mai mic decât va fi ilustrată prin exemplul unui alt joc video în care vârsta jucătorului se verifică chiar la începutul jocului (imaginea 5.3.).

Imaginea 5.3. Exemplu privind utilizarea operatorului mai mic decât

În imaginea 5.3., în partea stângă, puteți vedea că jocul este destinat doar jucătorilor cu vârsta peste 13 ani. Prin urmare, jucătorului i se cere să introducă data nașterii. Data pe care o introduce utilizatorul se scade din data de astăzi. Diferența este vârsta utilizatorului. Dacă este mai mică decât 13, jucătorului îi va fi afișat un mesaj de eroare. Dacă vârsta utilizatorului nu este mai mică decât 13, jucătorul are voie să

intre în joc.

Identic s-a putut obține folosind operatorul mai mare decât. În acest caz, afișajele ar fi inversate.

Pe lângă operatorii mai mare decât și mai mic decât, la fel ca în matematică, există operatori mai mare decât și egal cu și mai mic decât și egal cu. Exemplu privind utilizarea acestora în Python arată astfel:

```
value1 = 2
value2 = 2

print(value1 <= value2)
print(value1 >= value2)
```

În limbajul de programare Java, exemplul anterior poate fi formulat în modul următor:

```
int value1 = 2;
int value2 = 2;

System.out.println(value1 <= value2);
System.out.println(value1 >= value2);
```

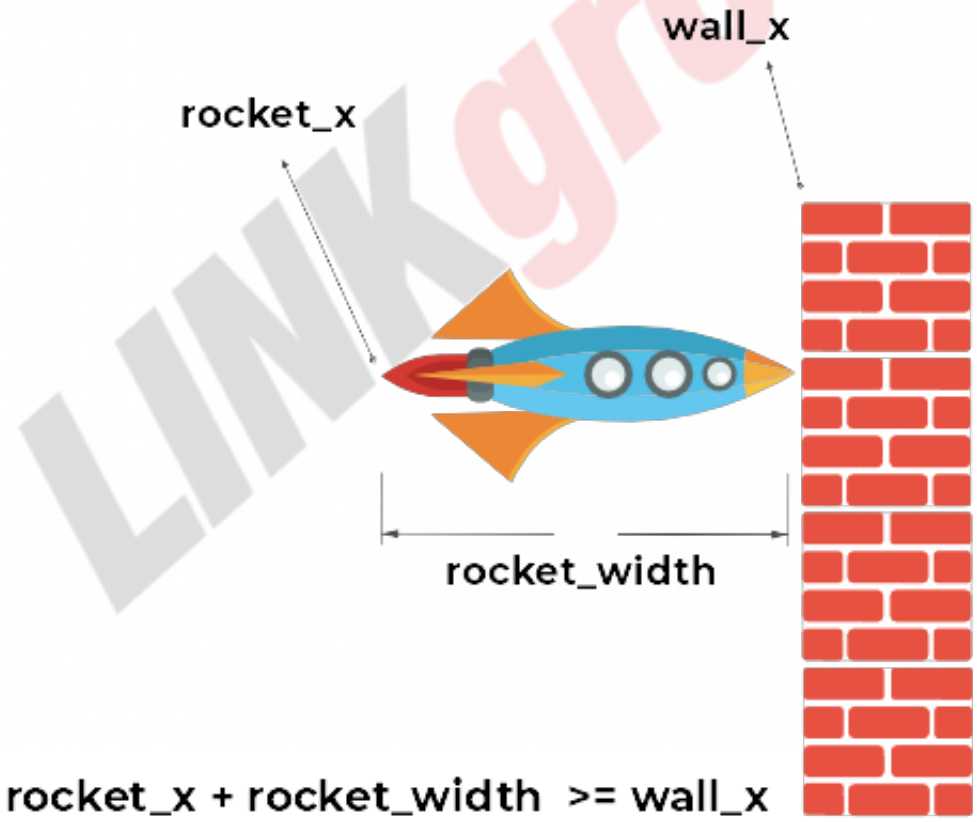
La ieșire se obțin următoarele rezultate:

```
True
True
```

Deoarece ambele valori sunt egale, ambele comparații sunt de fapt afirmații adevărate.

Exemple privind utilizarea operatorilor mai mare decât și egal cu și mai mic decât și egal cu

Un alt exemplu din lumea jocurilor video poate servi drept demonstrație a operatorului mai mare decât şi egal cu. Detectarea impactului unui proiectil (rachetă) asupra unui perete poate fi realizată tocmai de către operatorul mai mare decât şi egal cu (imaginea 5.4.).



Imaginea 5.4. Exemplu privind utilizarea operatorului mai mare decât şi egal cu

În imaginea 5.4. puteți vedea o rachetă care loveşte peretele. Este important să înţelegeţi că, în lumea jocurilor video, fiecare obiect

dispune de coordonate corespunzătoare pe display-ul dispozitivului utilizatorului. În cazul jocurilor 2D, sistemul de coordonate este format din două axe, x și y. Axa orizontală este axa x, în timp ce axa y este axa verticală. Pentru a detecta dacă racheta a avut contact cu peretele, este necesar să se verifice dacă coordonata sa x, mărită cu lățimea rachetei, este mai mare decât coordonata x a peretelui. Acest lucru se întreprinde în exemplul din imaginea 5.4. Pentru formarea expresiei, se folosește operatorul mai mare decât și egal cu.

Același lucru se poate realiza cu operatorul mai mic decât și egal cu:

```
wall_x <= rocket_x + rocket_width
```

Acum se verifică dacă coordonata x a peretelui este egală sau mai mică decât coordonata x a rachetei, mărită cu lățimea rachetei. Practic, efectul va fi identic, dar de data aceasta se realizează folosind un operator diferit.

Operatori logici

Operatorii logici se utilizează pentru a se determina legătura logică dintre variabile, respectiv valori. Adesea se utilizează în combinație cu operatorii de comparație tocmai prezentaţi.

Operatorii logici pot fi utilizați pentru a crea diverse expresii logice complexe, care trebuie să aibă întotdeauna `true` sau `false` ca valoare finală. Tabelul 5.4. prezintă operatorii logici.

Operator	Descriere	Exemplu	Rezultat
&&	AND	`(6 < 11 && 5 > 1)`	true
		`(6 < 5 && 5 > 1)`	false
||	OR	`(6 < 5 || 5 > 1)`	true
!	NOT	`!(6==6)`	false

Tabela 5.4. Logički operatori

Analizând exemplele de utilizare a operatorilor logici, se pot concluziona multe lucruri. Operatorul AND necesită îndeplinirea tuturor condițiilor de mai sus pentru a produce valoarea `true`. În caz contrar, produce valoarea `false`. Tabelul 5.5. ilustrează diferite situații de utilizare a operatorului logic AND.

Expresie	Rezultat
`true && true`	`true`
`false && true`	`false`
`true && false`	`false`
`false && false`	`false`

Tabelul 5.5. Operatorul logic AND

Pe de altă parte, operatorul OR necesită îndeplinirea doar a uneia dintre condiții pentru a produce `true` ca valoare finală (tabelul 5.6.).

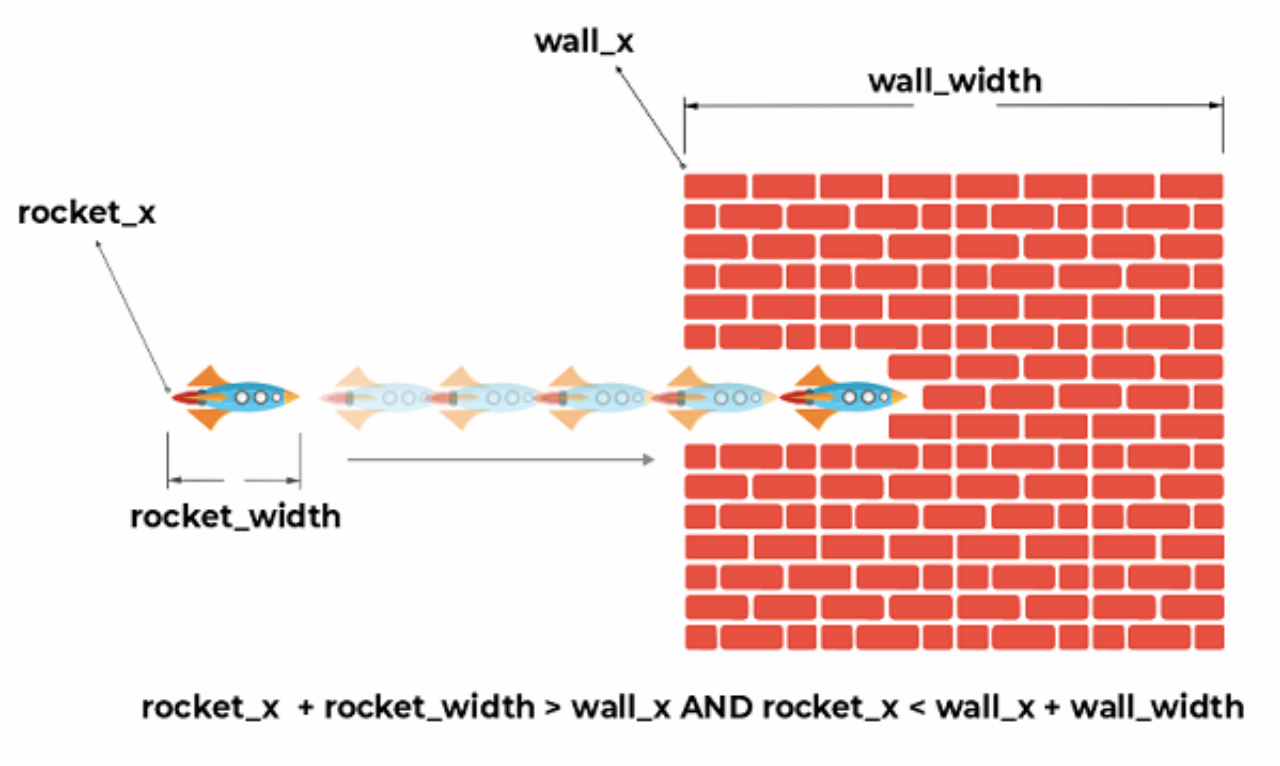
Expresie	Rezultat
`true || true`	`true`
`false || true`	`true`
`true || false`	`true`
`false || false`	`false`

Tabela 5.6. Operatorul logic OR

Exemple privind utilizarea operatorului logic AND

Exemplul anterior cu racheta și peretele ne va servi pentru a demonstra utilizarea operatorului logic AND. Întrebarea la care trebuie să se răspundă este: cum să determinați dacă racheta este în perete?

Imaginea 5.5. oferă răspuns la întrebare.



Imaginea 5.5. Exemplu privind utilizarea operatorului AND

În imaginea 5.5. puteți vedea racheta care trece prin perete. Folosind expresia din partea de jos a imaginii, determină dacă racheta se află în perete. Figura centrală a unei astfel de expresii este operatorul `AND`, prin care, de fapt, se împreunează două comparații. În primul rând, se determină dacă coordonata x a rachetei, mărită cu lățimea sa, este mai mare decât coordonata x a peretelui. În acest fel, se determină dacă vârful rachetei a avut contact cu peretele. Este de fapt o comparație pe care am văzut-o deja în exemplul anterior. Totuşi, de data aceasta, expresia se extinde cu o altă comparație după operatorul `AND`. Se determină dacă coordonata x a rachetei este mai mică decât coordonata x a peretelui, mărită cu lățimea acestuia. În acest mod, se determină dacă vârful rachetei a părăsit peretele. Folosirea operatorului AND înseamnă că ambele expresii (cea din stânga și cea dreapta) trebuie să producă valoarea `true` pentru ca întreaga expresie să fie `true`.

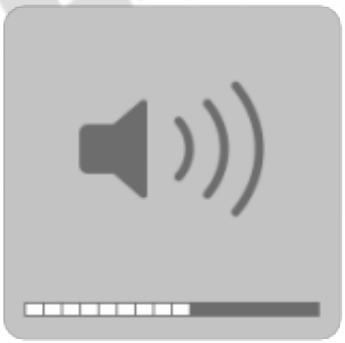
Operatorul NOT este un operator de negație, care transformă valoarea `true` în `false`, iar cea `false` în `true`, ceea ce este ilustrat în tabelul 5.7.

Expresie	Rezultat
`!true`	`false`
`!false`	`true`

Tabelul 5.7. Operatorul logic NOT

Exemplu privind utilizarea negației

Exemplu clasic de utilizare a operatorului de negație poate fi orice comutator care dispune de două stări - on și off. Un astfel de exemplu este butonul de control al volumului de pe sistemul de operare (imaginea 5.6.).



sound_enabled

!sound_enabled

Imaginea 5.6. Exemplu privind utilizarea operatorului NOT

În imaginea 5.6. este prezentat butonul de control al volumului. Utilizatorul are opțiunea de a opri complet sunetul făcând clic pe buton. Următorul clic reactivează sunetul. Cu alte cuvinte, cu fiecare clic ulterior, cele două stări (on-off) se schimbă.

Dacă constatăm că variabila `sound_enabled` se foloseşte pentru a stoca informații despre faptul dacă sunetul este activat sau nu, putem spune că, la fiecare clic pe butonul afișat, operatorul de negație poate fi folosit pentru a inversa valoarea acestuia.

Notă

Operatorii logici au o valoare de utilizare deosebit de importantă atunci când se creează condiții pentru controlul fluxului de execuție a codului. Controlul fluxului de execuție a programului va fi subiectul următoarei lecţii.

Exerciţii

```
public class JavaProgram
{

 public static void main(String[] args)
 {

  //todo

 }

}
```

Exerciţiul 1

Este necesar să se creeze un program care adună două numere reprezentate în două variabile. Primul număr este reprezentat de variabila x, cu valoarea 1, în timp ce cea de-a doua variabilă y are valoarea 2. Variabilei z este necesar să i se atribuie rezultatul adunării acestor două variabile. Rezultatul trebuie imprimat la ieşire.

Exerciţiul 2

Este necesar să se creeze un program care calculează aria cercului cu raza 5. Rezultatul trebuie imprimat 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.