INFORMATICĂ · CLASA A IX-A · STRUCTURI DE CONTROL · LECȚIA 4

Structura alternativă: if, else, switch

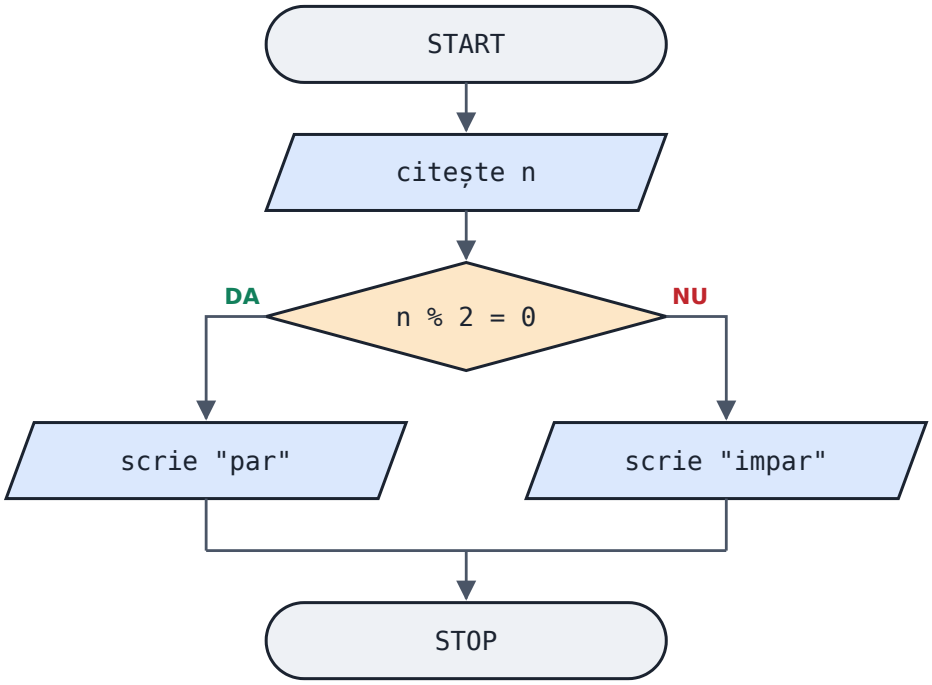
Durată: 60 min Nivel: toate profilurile Înainte: lecțiile ix-03

CE VEI ȘTI DUPĂ LECȚIE

- Scrii decizii simple și compuse cu if / else.
- Folosești if-uri imbricate și lanțuri else if fără să încurci ramurile.
- Construiești condiții compuse corecte cu &&, || și !
- Folosești switch pentru alegeri între mai multe valori.
- Determini maximul / minimul a 2–3 valori și ordonezi trei numere.

1. Programul ia decizii

Până acum programele tale mergeau „drept": citesc, calculează, afișează. Dar viața (și problemele) cer decizii: „dacă numărul e par, scrie *par*, altfel scrie *impar*". O decizie are mereu trei părți: o **întrebare** (condiția), o ramură pentru **DA** și, uneori, o ramură pentru **NU**.



Se parcurge o singură ramură: DA sau NU

Decizia cu două ramuri

Condiția este o expresie care are valoarea **adevărat** (în C++: orice valoare diferită de 0, de obicei 1) sau **fals** (0). De aceea tot ce ai învățat la lecția precedentă despre `<`, `==`, `&&`, `||`, `!` se folosește aici.

```cpp
if (conditie)
{
    // ramura DA
}
else
{
    // ramura NU (poate lipsi)
}
```

Pseudocod ↔ C++:

```
┌dacă n % 2 = 0 atunci          if (n % 2 == 0)
│   scrie "par"                      cout << "par";
│altfel                         else
│   scrie "impar"                    cout << "impar";
└■
```

Urmărirea unei decizii (tabel)

Pentru `n = 7`, apoi `n = 12`:

n	`n % 2`	`n % 2 == 0`	ramura aleasă	se afișează
7	1	fals	NU	`impar`
12	0	adevărat	DA	`par`

Observă: **din cele două ramuri se execută exact una**. Niciodată ambele, niciodată niciuna (când există `else`).

DE CE AȘA?

De ce `n % 2 == 0` și nu `n / 2`? Restul împărțirii la 2 spune dacă numărul „se împarte exact". `n / 2` ar da câtul, care nu răspunde la întrebarea „e par?". Pentru orice întrebare de felul „se divide cu k?", scrii `n % k == 0`.

ATENȚIE — GREȘELI FRECVENTE

Fără acolade, `if` controlează **doar prima** instrucțiune de după el:

```cpp
if (x > 0)
    cout << "pozitiv";
    cout << "!";        // se execută MEREU, indentarea nu contează pentru compilator
```

Regula simplă: **pune mereu acolade** dacă ai mai mult de o instrucțiune.

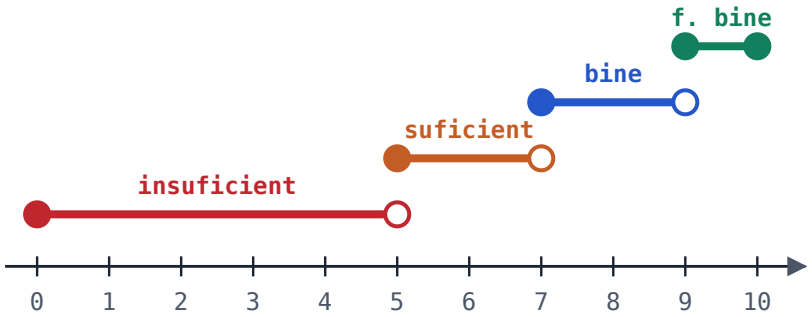
CAPCANĂ

`if (x > 0);` — punctul și virgulă de după paranteză termină `if`-ul. Blocul de după se execută întotdeauna. La fel, `if (x = 5)` **atribuie** 5 lui x (și condiția e mereu adevărată); comparația se scrie `==`.

2. Lanțul `else if` (mai multe cazuri)

Când ai mai mult de două cazuri, pui deciziile una după alta:

```
if (nota >= 9)
    cout << "foarte bine";
else if (nota >= 7)
    cout << "bine";
else if (nota >= 5)
    cout << "suficient";
else
    cout << "insuficient";
```



ercul plin = capăt inclus, cercul gol = capăt exclus (nota 9 e „foarte bine’

Lanțul else if pe axa notelor

Condițiile se verifică **de sus în jos** și se execută **doar prima** ramură adevărată; restul se sar.

nota	`>= 9`	`>= 7`	`>= 5`	rezultat
10	**da**	(nu se mai testează)		foarte bine
8	nu	**da**		bine
5	nu	nu	**da**	suficient
3	nu	nu	nu	insuficient (`else`)

DE CE AȘA?

De ce nu scriem `nota >= 7 && nota < 9` la a doua ramură? Nu e nevoie: dacă am ajuns acolo, prima condiție a fost falsă, deci știm deja că `nota < 9` . Fiecare ramură „moștenește” negarea celor de deasupra. Ordinea contează: dacă am începe cu `nota >= 5` , un 10 ar fi „suficient”.

3. Maximul a trei numere

Ideea (fără cod): ținem minte „cel mai mare văzut până acum” și îl comparăm pe rând cu fiecare număr nou.

Pentru `a = 4, b = 9, c = 6` :

pas	operația	mx
1	`mx = a`	4
2	`b > mx` ? 9 > 4 da → `mx = b`	9
3	`c > mx` ? 6 > 9 nu	9

```
int mx = a;
if (b > mx) mx = b;
if (c > mx) mx = c;
```

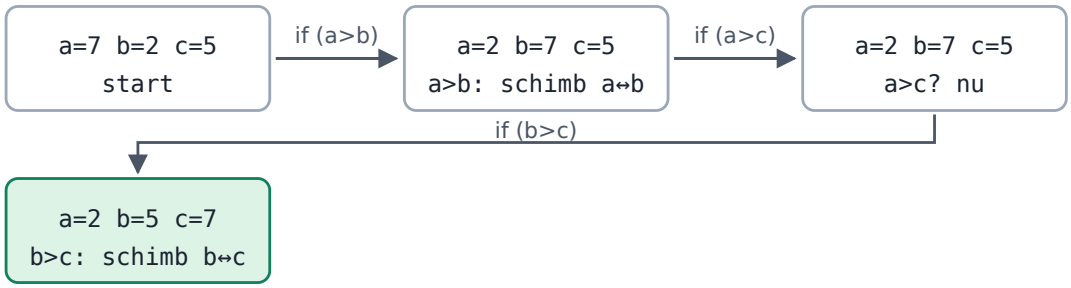
DE CE AȘA?

De ce varianta cu maxim parțial și nu `if (a >= b && a >= c) ... else if ...` ? Cea cu condiții compuse merge pentru 3 numere, dar pentru 4 ai 4 condiții cu 3 comparații fiecare. Varianta cu `mx` adaugă doar **un rând** pentru fiecare număr nou și va deveni, în lecțiile cu vectori, o buclă. Reține tiparul: *initializez cu primul, compar cu fiecare următor*.

Varianta cu condiții compuse (bună pentru a exersa `&&`):

```
if (a >= b && a >= c) mx = a;
else if (b >= a && b >= c) mx = b;
else mx = c;
```

Ordonarea crescătoare a trei numere



Ordonarea lui 7, 2, 5

```
int aux;
if (a > b) { aux = a; a = b; b = aux; }   // acum a <= b
if (a > c) { aux = a; a = c; c = aux; }   // acum a este minimul
if (b > c) { aux = b; b = c; c = aux; }   // acum a <= b <= c
cout << a << ' ' << b << ' ' << c;
```

Urmărire pentru `a = 7, b = 2, c = 5`:

după instrucțiunea	a	b	c
(început)	7	2	5
`if (a > b)` → schimb	2	7	5

după instrucțiunea	a	b	c
`if (a > c)` → nu	2	7	5
`if (b > c)` → schimb	2	5	7

DE CE AȘA?

De ce trei comparații, în această ordine? După primele două, `a` este sigur cel mai mic (a fost comparat cu ambii). Rămâne doar de pus în ordine perechea `b, c`. Dacă schimbi ordinea if-urilor, rezultatul poate fi greșit; testează-l singur pe `3 2 1`.

SFAT

Interschimbarea a două variabile se face cu o a treia („paharul gol"): `aux = a; a = b; b = aux;`. Dacă scrii `a = b; b = a;`, ambele ajung egale cu b. În C++ există și `swap(a, b);`.

4. Condiții compuse — tipare utile

Vreau să verific...	Condiția
x în intervalul [a, b]	`x >= a && x <= b`
x în afara intervalului [a, b]	`x < a || x > b`
n are exact două cifre	`n >= 10 && n <= 99`
a, b, c pot fi laturile unui triunghi	`a + b > c && a + c > b && b + c > a`
an bisect	`(an % 4 == 0 && an % 100 != 0) || an % 400 == 0`
exact unul dintre a, b este par	`(a % 2 == 0) != (b % 2 == 0)`
toate trei egale	`a == b && b == c`

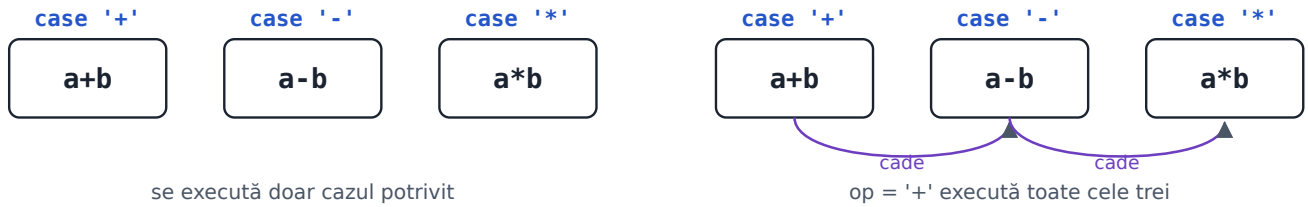
ATENȚIE — GREȘELI FRECVENTE

Nu există „interval" în C++: `1 <= x <= 9` **se compilează**, dar nu face ce crezi. Se evaluează `(1 <= x) <= 9`; parantezele dau 0 sau 1, iar 0 și 1 sunt oricum `<= 9`. Condiția e mereu adevărată! Scrie `x >= 1 && x <= 9`.

5. `switch` — alegere după valoare

Când alegi între mai multe **valori exacte** ale aceleiași variabile, `switch` e mai clar decât un lanț de `else if`:

```cpp
#include <iostream>
using namespace std;
int main()
{
    char op;
    int a, b;
    cin >> a >> op >> b;
    switch (op)
    {
        case '+': cout << a + b; break;
        case '-': cout << a - b; break;
        case '*': cout << a * b; break;
        case '/':
            if (b != 0) cout << a / b;
            else cout << "impartire la zero";
            break;
        default: cout << "operatie necunoscuta";
    }
}
```



switch: rolul lui break

- `switch` funcționează doar cu valori **întregi** sau `char` (nu cu intervale, nu cu `double`).
- `default` se execută dacă nicio valoare nu se potrivește (ca `else`).
- Fără `break`, execuția „cade” în cazul următor.

DE CE AȘA?

De ce există `break`? Eticheta `case` doar marchează de unde începe execuția; nu marchează unde se termină. `break` spune „am terminat, ies din switch”. Căderea voită e uneori utilă: `case 'a': case 'e': case 'i': cout << "vocala"; break;` grupează mai multe valori.

Urmărire pentru intrarea `8 * 3`: `op` este `'*'` → se sare la `case '*'` → se afișează 24 → `break` → ieșire.

6. Exemplu rezolvat: an bisect (`#177 bisect`)

Un an este bisect dacă se divide cu 4, dar nu cu 100, **sau** dacă se divide cu 400.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int an;
    cin >> an;
    if ((an % 4 == 0 && an % 100 != 0) || an % 400 == 0)
        cout << "DA";
    else
        cout << "NU";
    return 0;
}
```

an	`% 4 == 0`	`% 100 != 0`	prima paranteză	`% 400 == 0`	rezultat
2024	da	da	**da**	nu	DA
1900	da	nu	nu	nu	NU
2000	da	nu	nu	**da**	DA
2023	nu	da	nu	nu	NU

Intrare / ieșire:

Intrare	Ieșire
`2024`	`DA`
`1900`	`NU`
`2000`	`DA`

DE CE AȘA?

De ce parantezele în jurul lui `&&` ? `&&` are prioritate mai mare decât `||`, deci ar merge și fără, dar parantezele arată intenția: „(4 și nu 100) sau 400". Cititorul (și tu peste o lună) vede imediat regula. (Verifică pe pbinfo exact ce mesaje cere enunțul.)

7. Testarea unui program cu decizii

Pentru fiecare ramură alege măcar un test care să intre pe ea, plus **cazurile de la limită**:

- valori egale (`a == b`), zero, numere negative (dacă sunt permise);
- capetele intervalelor (`x == a` , `x == b`).

Exemplu: pentru `nota` testezi 4, 5, 6, 7, 8, 9, 10. Limitele (4/5, 6/7, 8/9) sunt acolo unde apar greșelile `>` în loc de `>=` .

8. Greșeli frecvente

Greșeala	Ce se întâmplă	Corect
`if (x = 5)`	atribuie, condiția e mereu adevărată	`if (x == 5)`
`if (x > 0);`	blocul se execută mereu	fără `;`
`1 <= x <= 9`	mereu adevărat	`x >= 1 && x <= 9`
`else if` în loc de două `if` independente	a doua condiție nu se mai verifică	alege după ce vrei: cazuri exclusive sau independente
`case` fără `break`	„cade” în următorul caz	pune `break;`

9. La BAC

> **LA BAC**
>
> Subiectul I are mereu o grilă cu **condiții** („indicați expresia care are valoarea 1 dacă și numai dacă...”) și una cu **negarea** unei condiții (De Morgan). Metoda sigură: alege 3 valori de test (în interval, la capăt, în afară) și verifică fiecare variantă. Subiectul al II-lea cere adesea să scrii un `if` care alege între două prelucrări; ține minte tiparul cu `mx`.

Exemplu: care condiție înseamnă „x aparține lui (5, 8]”?

Test	x=5	x=6	x=8	x=9
`x > 5 && x <= 8`	0	1	1	0
`x >= 5 && x < 8`	1	1	0	0

Intervalul (5, 8] exclude 5 și include 8: potrivită este prima. Capetele au rezolvat grila.

10. Exersează pe hârtie

Răspunsurile sunt la finalul lecției, după probleme: încearcă întâi singur.

1. Ce afișează secvența pentru `x = 7`, apoi pentru `x = 12`?

```
if (x % 3 == 0 || x > 10) cout << "A";
else if (x % 2 == 1) cout << "B";
else cout << "C";
```

2. Scrie o singură instrucțiune `if` care afișează „DA” dacă numărul de 3 cifre `n` are cifrele în ordine strict crescătoare.

3. Rescrie fără `!`: `!(a > 5 || b == 0)`.

4. Ce valoare are `y` la final?

```
int x = 4, y = 0;
if (x > 3) y = y + 1;
if (x > 2) y = y + 2;
else y = y + 4;
if (x > 5) y = y + 8;
```

5. Care dintre condiții înseamnă „x nu este în intervalul [2, 6]”?

a) `x < 2 && x > 6`

b) `x < 2 || x > 6`

c) `!(x >= 2 && x <= 6)`

d) `x <= 2 || x >= 6`

(Pot fi mai multe corecte.)

6. Ce afișează secvența pentru `k = 2`?

```
switch (k)
{
    case 1: cout << "a";
    case 2: cout << "b";
    case 3: cout << "c"; break;
    default: cout << "d";
}
```

7. Scrie secvența care afișează, pentru un număr `n`, „negativ”, „zero” sau „pozitiv”.

8. Ce afișează `cout << (a > b ? a : b);` pentru `a = 3, b = 8`? Ce calculează expresia în general?

11. Mai departe (pentru cine vrea mai mult)

Evaluarea „scurtcircuit”. În `a && b`, dacă `a` e fals, `b` nu se mai calculează. De aceea `if (b != 0 && a / b > 2)` nu se împarte niciodată la zero. Ordinea condițiilor într-un `&&` sau `||` este o unealtă de protecție, nu doar de stil.

Decizii fără `if`. Valoarea unei comparații este 0 sau 1, deci `mx = a * (a >= b) + b * (a < b);` calculează maximul fără `if`. Nu este mai clar, dar apare în cod de performanță (fără ramificări, procesorul merge mai repede) și în grile.

Arbori de decizie. Un lanț de if-uri imbricate este un arbore de decizie. Cu aceeași idee funcționează sistemele de recomandare simple, filtrele de spam cu reguli și logica personajelor din jocuri. Încearcă: desenează pe hârtie arborele pentru „ce să port azi?” (ploaie? frig? vânt?) și apoi scrie-l ca if-uri.

Cazuri la limită în producție. Majoritatea bug-urilor reale din decizii sunt la limite (`<` vs `<=`). Un obicei bun: scrie întâi testele de la capete, apoi codul.

Rezumat

- `if (cond) {...} else {...}`: se execută exact o ramură; condiția e adevărată dacă e diferită de 0.
- `else if` verifică de sus în jos și oprește la prima condiție adevărată.

- Maximul: `mx = a; if (b > mx) mx = b; ...` — tiparul „maxim parțial”.
- Interval: `x >= a && x <= b`; nu `a <= x <= b`.
- `switch` merge pe întregi/`char`; fiecare `case` are nevoie de `break`.
- Testează ramurile și capetele intervalelor.

Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] **#105** **max2** — ușoară
  Maximul a două numere — două variante: if și ?:
  https://www.pbinfo.ro/probleme/105/max2
- [ ] **#109** **paritate** — ușoară
  https://www.pbinfo.ro/probleme/109/paritate
- [ ] **#9** **maxmin** — ușoară
  Maximul și minimul a trei numere.
  https://www.pbinfo.ro/probleme/9/maxmin
- [ ] **#177** **bisect** — ușoară
  Condiție compusă: divizibil cu 4 și nu cu 100, sau divizibil cu 400.
  https://www.pbinfo.ro/probleme/177/bisect

Temă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] **#106** **minim3** — ușoară
  https://www.pbinfo.ro/probleme/106/minim3
- [ ] **#167** **semn** — ușoară
  https://www.pbinfo.ro/probleme/167/semn
- [ ] **#179** **triunghi** — ușoară
  Inegalitatea triunghiului pentru toate cele trei laturi.
  https://www.pbinfo.ro/probleme/179/triunghi
- [ ] **#446** **cifmax3** — ușoară
  https://www.pbinfo.ro/probleme/446/cifmax3
- [ ] **#469** **interval2** — ușoară
  Atenție: a și b pot fi date în orice ordine.
  https://www.pbinfo.ro/probleme/469/interval2
- [ ] **#453** **distincte1** — ușoară
  https://www.pbinfo.ro/probleme/453/distincte1

- [ ] #1303 **calculator** — ușoară
  Bun pentru switch pe caracterul operației.
  https://www.pbinfo.ro/probleme/1303/calculator

- [ ] #832 **nota** — ușoară
  https://www.pbinfo.ro/probleme/832/nota

Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #1625 **ec2** — ușoară
  Ecuația de gradul al doilea: toate cazurile (Δ < 0, = 0, > 0, a = 0).
  https://www.pbinfo.ro/probleme/1625/ec2

- [ ] #793 **sumprod** — ușoară
  https://www.pbinfo.ro/probleme/793/sumprod

Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #7 **Exercițiul #7**
  interval (5,8]: ce condiție îl descrie (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=7

- [ ] #15 **Exercițiul #15**
  echivalența unei condiții compuse (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=15

- [ ] #18 **Exercițiul #18**
  negarea lui && : De Morgan (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=18

- [ ] #29 **Exercițiul #29**
  negarea unei condiții cu trei relații (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=29

- [ ] #61 **Exercițiul #61**
  !((x>=5)&&(x<10)) (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=61

- [ ] #12 **Exercițiul #12**
  expresie care dă minimul a două numere (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=12

- [ ] #549 **Exercițiul #549**
  condiție compusă pe intervale (Bac 2011)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=549

- [ ] #524 **Exercițiul #524**
  condiție pe cifrele unui număr (Bac 2012)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=524

- [ ] #709 **Exercițiul #709**
  echivalența cu o negație (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=709

- [ ] #660 **Exercițiul #660**
  număr cu exact trei cifre (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=660

- [ ] #1245 **Exercițiul #1245**
  instrucțiune if echivalentă cu x = -abs(y) (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1245

Citește și exersează în plus

- Structuri alternative https://www.pbinfo.ro/articole/70/structuri-alternative
- Operatorul condițional ? https://www.pbinfo.ro/articole/26006/operatorul-conditional
- Operații logice https://www.pbinfo.ro/articole/21315/operatii-logice
- Python Tutor pentru C++ — rulează un if pas cu pas și vezi ce ramură se alege
  https://pythontutor.com/cpp.html
- cppreference — instrucțiunea if — referința oficială (engleză), cu exemple scurte
  https://en.cppreference.com/w/cpp/language/if
- cppreference — instrucțiunea switch — include și explicația pentru „fall-through”
  https://en.cppreference.com/w/cpp/language/switch

Răspunsuri

Exersează pe hârtie

1. Pentru `x = 7`: `7 % 3 = 1` și `7 > 10` fals → condiția 1 e falsă; `7 % 2 == 1` adevărat → `B`. Pentru `x = 12`: `12 % 3 == 0` → `A`.

2. `if (n / 100 < n / 10 % 10 && n / 10 % 10 < n % 10) cout << "DA";` Cifrele sunt sute `n/100`, zeci `n/10%10`, unități `n%10`.

3. `a <= 5 && b != 0` (De Morgan: `!` pe `||` devine `&&` și se neagă fiecare relație).

4. `y = 3`. Primul if: y devine 1. Al doilea: `x > 2` adevărat → y = 3 (`else` nu se execută). Al treilea: `x > 5` fals.

5. b) și c). Varianta a) nu e niciodată adevărată (nu poate fi și mai mic ca 2, și mai mare ca 6). Varianta d) greșește la 2 și 6, care sunt în interval dar fac condiția adevărată.

6. `bc`. Se sare la `case 2`, se afișează `b`, nu e `break`, deci cade în `case 3`, afișează `c` și `break` oprește.

7.

```
if (n < 0) cout << "negativ";
else if (n == 0) cout << "zero";
else cout << "pozitiv";
```

8. Afișează `8`. Expresia `a > b ? a : b` calculează maximul dintre `a` și `b`.