FIȘĂ DE LUCRU · CLASA A XI-A · FIȘA 2 · ACOPERĂ LECȚIILE XI.3–XI.4

Recursivitate: stiva de apeluri, cifre, vectori, fill

Durată: 100 min Nivel: mate-info

CE VEI ȘTI DUPĂ LECȚIE

- Urmărești pe stivă orice funcție recursivă și spui exact ce afișează sau ce returnează (grila de la Subiectul I).
- Repari funcțiile recursive greșite: caz de bază, apel care nu se apropie, parametru prin valoare în loc de referință.
- Scrii recursiv prelucrări pe cifre, pe vectori și algoritmul de umplere (fill).

Ce trebuie să știi (o pagină)

Regula	Cod tipic	Ce reții
caz de bază + apel pe o problemă **mai mică**	`if (n == 0) return 1; return n * f(n - 1);`	fără caz de bază: *stack overflow*
fiecare apel are **propriile** variabile, pe stivă	`f(3)` așteaptă `f(2)`, care așteaptă `f(1)`	apelurile se termină în ordine inversă
cifrele: ultima `n % 10` + restul `n / 10`	`return n % 10 + s(n / 10);`	cazul de bază: `n == 0` sau `n < 10`
`cout` **înainte** de apel → la coborâre	`cout << n; f(n - 1);`	ordinea apelurilor
`cout` **după** apel → la urcare	`f(n - 1); cout << n;`	ordinea inversă
două apeluri → arbore de apeluri	`f(n - 1) + f(n - 2)`	multe calcule repetate
parametru prin **referință** = o singură variabilă comună	`void f(int n, int &cnt)`	prin valoare, fiecare apel are o copie
vector pe lungime / pe indice / pe interval	`f(a, n - 1) + a[n]`, `f(a, i + 1, n)`, `f(st + 1, dr - 1)`	cazul de bază: vector vid sau un element
fill: margine → verificare → **marcare** → vecini	`if (în afară sau a[i][j] != 1) return 0;` `a[i][j] = 2;`	marcarea se face **înainte** de apeluri

SFAT

Metoda sigură la grile: un tabel cu coloanele *apel* · *afișează înainte* · *apelează* · *afișează după* · *returnează*. Îl completezi de sus în jos (coborârea), apoi de jos în sus (urcarea). Nu citi codul „pe diagonală”.

Nivelul 1 · Prezice

★ citește și prezice

P1. Completează tabelul pentru funcția de mai jos, apoi spune **în cuvinte** ce face.

```
int f(int n) {
    if (n == 0) return 0;
    if (n % 2 == 0) return f(n / 10) * 10 + n % 10;
    return f(n / 10);
}
```

f(2468)	f(1357)	f(52103)	f(907)	f(8)

P2. Ce afișează apelul `f(3, 'a')`?

```
void f(int n, char c) {
    if (n == 0) { cout << '|'; return; }
    cout << c;
    f(n - 1, c + 1);
    cout << c;
}
```

P3. Fie `int f(int n) { if (n <= 1) return n; return f(n - 1) + 2 * f(n - 2); }`. Calculează `f(6)` **de jos în sus** (completează tabelul), apoi numără câte apeluri face programul pentru `f(6)`, inclusiv primul.

n	0	1	2	3	4	5	6
f(n)							

P4. Ce valoare are `f(30, 3)` și câte apeluri se fac în total?

```
int f(int a, int b) {
    if (a < b) return a;
    return f(a - b, b + 1);
}
```

P5. Vectorul global este `a = (3, 8, 1, 9, 4)`, indexat de la 1. Ce afișează `f(1, 5)`? Ce afișează dacă muți `if (i != j) cout << a[j] << ' ';` **înaintea** apelului și `cout << a[i] << ' ';` după el?

```
void f(int i, int j) {
    if (i <= j) {
        cout << a[i] << ' ';
        f(i + 1, j - 1);
        if (i != j) cout << a[j] << ' ';
    }
}
```

P6. Pe harta de mai jos se apelează `fill(1, 1)` din lecție (versiunea care marchează cu 2), cu vecinii încercați în ordinea **sus, jos, stânga, dreapta**. Scrie în fiecare celulă de pământ al câtelea a fost marcată.

	col. 1	col. 2	col. 3
linia 1	1	1	0
linia 2	1	1	1
linia 3	0	0	1

P7. Ce afișează `f(s, 0)` pentru `char s[] = "informatica";` ? Dar pentru `"bac"` ?

```
void f(char s[], int i) {
    if (s[i] == 0) return;
    if (s[i + 1] != 0) f(s, i + 2);
    cout << s[i];
}
```

P8. Funcția se apelează **pe ea însăși în interiorul propriului apel**. Cât este `f(9875)` ? Dar `f(99)` ? Ce calculează, în cuvinte?

```
int f(int n) { if (n < 10) return n; return f(f(n / 10) + n % 10); }
```

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Funcția trebuie să întoarcă produsul cifrelor lui `n` , dar `prod(234)` dă 0 (și orice alt apel la fel). De ce? Repară **doar cazul de bază**.

```
int prod(int n) { if (n == 0) return 0; return n % 10 * prod(n / 10); }
```

P10. Elevul vrea să numere cifrele 0 ale lui `n` . Pentru `10203` programul afișează `0` , nu `2` . Repară-l adăugând **un singur caracter**.

```
void numara(int n, int cnt) {
    if (n == 0) return;
    if (n % 10 == 0) cnt++;
    numara(n / 10, cnt);
}
int main() { int c = 0; numara(10203, c); cout << c; }
```

P11. *Puzzle cu rânduri amestecate.* Funcția verifică recursiv dacă `a[1..n]` este crescător. Pune rândurile în ordine; un rând este în plus.

```
A)      return crescator(a, n - 1);
B)  bool crescator(int a[], int n) {
C)      if (a[n - 1] > a[n]) return false;
D)      if (n == 0) return false;
E)  }
F)      if (n <= 1) return true;
```

P12. Completează golurile ca `fill` să întoarcă aria zonei cu **8 vecini** (și pe diagonală).

```
int di[] = {-1, -1, -1, 0, 0, 1, 1, 1}, dj[] = { _________________ };
int fill(int i, int j) {
    if (i < 1 || i > n || j < 1 || j > m || a[i][j] != 1) return ___;
    a[i][j] = ___;
    int arie = ___;
    for (int d = 0; d < 8; d++) arie += fill(______, ______);
    return arie;
}
```

P13. `sumaPare(n)` ar trebui să dea suma numerelor pare de la 2 la `n`. Pentru `n = 6` merge, dar pentru `n = 7` programul se blochează și apoi se oprește cu eroare. Explică ce se întâmplă pe stivă și repară funcția ca să meargă pentru orice `n ≥ 0`.

```
int sumaPare(int n) { if (n == 0) return 0; return n + sumaPare(n - 2); }
```

P14. Compilatorul dă un avertisment („control reaches end of non-void function"), iar rezultatul pentru `a = (3, 9, 2, 7)` nu este garantat. Ce lipsește?

```
int maxim(int a[], int n) {
    if (n == 1) return a[1];
    int m = maxim(a, n - 1);
    if (a[n] > m) return a[n];
}
```

Nivelul 3 · Scrie cod

★★ scrie cod

Toate funcțiile cerute sunt **recursive** (fără `for` / `while` în ele). Verifică toate testele.

P15. Scara. Urci o scară cu `n` trepte făcând pași de 1, 2 sau 3 trepte. În câte moduri poți ajunge sus? Teste: `1 → 1`, `3 → 4`, `5 → 13`, `10 → 274`.

P16. Oglinda. Scrie `int oglinda(int n, int r)` care întoarce numărul scris invers; în `main` apelezi `oglinda(n, 0)`. Teste: `1234 → 4321`, `1200 → 21`, `7 → 7`.

P17. Calculatorul vechi. Afișează `n` scris în baza `b` (2 ≤ b ≤ 10), cu o funcție `void baza(int n, int b)`. Teste: `13 2 → 1101`, `100 8 → 144`, `0 5 → 0`.

P18. Termometrul. Se dau `n` și temperaturile `t[1..n]`. Numără recursiv zilele în care s-a făcut mai cald decât în ziua precedentă. Teste: `7` / `3 5 4 6 8 8 2` → `3`; `1` / `10` → `0`; `4` / `1 2 3 4` → `3`.

P19. Lacurile. Harta are `1` = uscat și `0` = apă. Un **lac** este o zonă de apă (4 vecini) care **nu atinge marginea** hărții. Câte lacuri sunt?

	test 1	test 2	test 3

intrare	`5 6`
	`1 1 1 1 1 0`
	`1 0 0 1 1 0`
	`1 1 1 1 0 1`
	`1 0 1 1 1 1`
	`1 1 1 0 0 1`
		`3 3`
		`1 1 1`
		`1 0 1`
		`1 1 1`
			`2 2`
			`0 0`
			`0 0`

ieșire	`3`	`1`	`0`

P20. Coada de la casă, invers. Se citesc numere până la `0` (exclusiv). Afișează numerele **pare**, în ordinea **inversă** citirii, **fără vector**. Teste: `3 8 5 2 10 0` → `10 2 8`; `1 3 0` → nimic; `4 0` → `4`.

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Detectivul invers. Pentru funcția `f` din P1, **câte** numere de trei cifre au `f(n) = 8`? Raționează pe cazuri, fără să le scrii pe toate. (Atenție la cifra 0!)

P22. Înmulțirea țăranului rus. Funcția `int inm(int a, int b) { if (b == 0) return 0; return a + inm(a, b - 1); }` face `b + 1` apeluri. Scrie o variantă care folosește ideea de la `putere` (`b / 2`) și explică de ce e corectă. Câte apeluri face pentru `b = 1000`? Dar pentru `b = 1 000 000`?

P23. Câte apeluri face Fibonacci? Pentru `int f(int x) { if (x <= 1) return x; return f(x - 1) + f(x - 2); }`, notează cu `A(n)` numărul de apeluri făcute pentru `f(n)`. Completează `A(0), …, A(7)`, găsește o relație între `A(n)`, `A(n − 1)` și `A(n − 2)`, apoi, știind că `f(21) = 10946`, prezice `A(20)` (indiciu: compară `A(n)` cu `f(n + 1)`).

P24. Contraexemplul. Un elev a scris cifra minimă astfel: `int cifMin(int n) { if (n == 0) return 9; return min(n % 10, cifMin(n / 10)); }`. Colegul spune că e greșit. Există un număr natural pentru care funcția greșește? Găsește **toate** astfel de valori și explică de ce celelalte merg.

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] #823 **sumcifrec** — ușoară
  https://www.pbinfo.ro/probleme/823/sumcifrec
- [ ] #829 **afisarerec** — ușoară
  https://www.pbinfo.ro/probleme/829/afisarerec
- [ ] #1843 **fsumvecrec** — ușoară
  https://www.pbinfo.ro/probleme/1843/fsumvecrec
- [ ] #837 **fill** — ușoară
  https://www.pbinfo.ro/probleme/837/fill

Temă pentru acasă (pbinfo)

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

- [ ] #824 **cifmaxrec** — ușoară
  https://www.pbinfo.ro/probleme/824/cifmaxrec
- [ ] #1752 **nr-aparitii-rec** — ușoară
  https://www.pbinfo.ro/probleme/1752/nr-aparitii-rec
- [ ] #1511 **fcautarerec** — ușoară
  https://www.pbinfo.ro/probleme/1511/fcautarerec

- [ ] #4042 **b16-rec** — ușoară
  https://www.pbinfo.ro/probleme/4042/b16-rec
- [ ] #840 **croco** — medie
  https://www.pbinfo.ro/probleme/840/croco
