

TEST DE EVALUARE

1. **Ce face funcția pow în C++?**
 - a) Calculează radicalul unui număr.
 - b) Calculează puterea unui număr (adică numărul ridicat la o putere dată).
 - c) Calculează media unui șir de numere.
2. **Ce face funcția sqrt în C++?**
 - a) Calculează pătratul unui număr.
 - b) Calculează suma a două numere.
 - c) Calculează radicalul pătrat al unui număr.
3. **Care dintre sintaxele de mai jos este cea corectă pentru a calcula 2 la puterea 3 în C++?**
 - a) pow(2, 3)
 - b) sqrt(2, 3)
 - c) pow(3, 2)
4. **Ce tip de rezultat returnează funcția sqrt în C++?**
 - a) Un caracter
 - b) Un număr întreg sau în virgulă mobilă (float sau double), în funcție de valoarea citită.
 - c) Un număr complex
5. **Ce trebuie să incluzi în programul tău C++ pentru a utiliza funcțiile pow, sqrt, și alte funcții matematice?**
 - a) #include <iostream>
 - b) #include <cmath>
 - c) #include <math.h>
6. **Ce face operatorul / în C++ atunci când este aplicat între două numere întregi?**
 - a) Returnează restul împărțirii celor două numere.
 - b) Returnează rezultatul împărțirii întregi a celor două numere.
 - c) Returnează rezultatul în virgulă mobilă al împărțirii.
7. **Ce face operatorul % în C++ atunci când este aplicat între două numere întregi?**
 - a) Returnează împărțirea între numerele întregi.
 - b) Returnează restul împărțirii celor două numere întregi.
 - c) Returnează valoarea absolută a diferenței dintre cele două numere.
8. **Care din secvențele următoare este incorectă pentru a afla maximul dintr-un șir de n numere întregi, introduse de la tastatură, mai mici decât 1 000 000, unde $1 \leq n \leq 20$**

a)	b)	c)
<pre>#include <iostream> using namespace std; int main() { int v[100]; int n, i, max; max=0; cout<<"Nr de valori n="; cin>>n; for (i=0; i<=n-1; i++) cin>>v[i]; for (i=0; i<=n-1; i++) { if(max<v[i]) max=v[i]; } cout<<"Maximul ="<<max; return 0; }</pre>	<pre>#include <iostream> using namespace std; int main() { int v[100]; int n, i, max; cout<<"Nr de valori n="; cin>>n; for (i=0; i<=n-1; i++) cin>>v[i]; max=v[0]; for (i=1; i<=n-1; i++) { if(max<v[i]) max=v[i]; } cout<<"Maximul ="<<max; return 0; }</pre>	<pre>#include <iostream> using namespace std; int main() { int v[100]; int n, i, max; cout<<"Nr de valori n="; cin>>n; for (i=0; i<=n-1; i++) cin>>v[i]; max=v[0]; for (i=1; i<=n-1; i++) { if(max>v[i]) max=v[i]; } cout<<"Maximul ="<<max; return 0; }</pre>

9. Media aritmetică a elementelor unui vector se calculează folosind formula:

$$m_a = \frac{x_1 + x_2 + \dots + x_n}{n}$$

Pentru următoarea situație practică: *La o stație meteo se fac zilnic măsurători ale temperaturii exterioare într-o lună de vară. Realizați un program care să afișeze care este temperatura medie din luna respectivă și numărul de zile în care temperatura a depășit pragul mediu al lunii.*

Pentru a calcula media aritmetică (**ma**) și a afișa numărul de zile în care se depășește temperatura medie, folosim secvența:

a)	b)	c)
for (i=0; i<n; i++) s=s+v[i]; ma=n/s; cout<<"Media="<<ma<<endl;	for (i=0; i<n; i++) s=s+v[i]; ma=n/s; cout<<"Media="<<ma<<endl;	for (i=0; i<n; i++) s=s+v[i]; ma=s/n; cout<<"Media="<<ma<<endl;
for (i=0; i<n; i++) if (v[i]>ma) nr++; cout<<"Nr. valori mai mari decat media:"<<nr;	for (i=0; i<n; i++) if (v[i]<ma) nr++; cout<<"Nr. valori mai mari decat media:"<<nr;	for (i=0; i<n; i++) if (v[i]>ma) nr++; cout<<"Nr. valori mai mari decat media:"<<nr;

10. Media pătratică (numită și media quadratică sau rădăcina mediei pătrate) este o măsură statistică care se folosește pentru a determina valoarea medie a unui set de numere, punând accent pe valorile mai mari. Este definită ca rădăcina pătrată a mediei aritmetice a pătratelor valorilor dintr-un șir.

Dacă a și b sunt numere reale pozitive media pătratică a acestora este:

$$m_p = \sqrt{\frac{a^2 + b^2}{2}}$$

Generalizare: pentru n numere reale pozitive $x_1, x_2, \dots, x_n$ formula mediei pătratice este:

$$m_p = \sqrt{\frac{x_1^2 + x_2^2 + \dots + x_n^2}{n}}$$

În C++ calculul mediei pătratice (**mp**) se face astfel:

a)	b)	c)
for (i=0; i<n; i++) s=s+v[i]*v[i]; mp=sqrt(s/n); cout<<"Media pătratică ="<<mp;	for (i=0; i<n; i++) s=s+v[i]; mp=sqrt(s/n); cout<<"Media pătratică ="<<mp;	for (i=0; i<n; i++) s=s+v[i]*v[i]; mp=pow(s,n); cout<<"Media pătratică ="<<mp;

11. Media armonică (m_h , unde h- provine de la harmonic) a elementelor $x_1, x_2, \dots, x_n$ ale unui șir de n numere reale pozitive este:

$$m_h = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \dots + \frac{1}{x_n}}$$

În C++ calculul mediei armonice (**mh**) se face astfel:

a)	b)	c)
for (i=0; i<n; i++) s=s+v[i]; mh=n/s; cout<<"Media armonica: "<<mh;	for (i=0; i<n; i++) s=s+1/v[i]; mh=n/s; cout<<"Media armonica: "<<mh;	for (i=0; i<n; i++) s=s+1/v[i]; mh=s/n; cout<<"Media armonica: "<<mh;

- 12. Media geometrică** are sens numai pentru numere reale pozitive. Media geometrică pentru n numere:

$$M_g = \sqrt[n]{a_1 * a_2 * \dots * a_n}$$

Pentru a o folosi într-un program C++, formula se poate rescrie astfel:

$$M_g = (a_1 * a_2 * \dots * a_n)^{1/n}$$

În C++, calculul mediei geometrice se face astfel:

a)	b)	c)
<pre>p=0; for (i=0; i<n; i++) p=p*v[i]; cout<< pow(p,1.0/n);</pre>	<pre>p=1; for (i=0; i<n; i++) p=p*v[i]; cout<<sqrt(p);</pre>	<pre>p=1; for (i=0; i<n; i++) p=p*v[i]; cout<< pow(p,1.0/n);</pre>

- 13. Cum putem verifica dacă toate numerele dintr-un vector sunt prime?**

- Iterăm prin fiecare număr și verificăm dacă fiecare este mai mare decât 1 și nu are divizori între 2 și $\sqrt{n}$.
- Verificăm doar primul și ultimul număr.
- Adunăm toate numerele și verificăm suma.

- 14. Din secvențele următoare care este cea corectă pentru a calcula suma elementelor prime dintr-un vector de n elemente citite de la tastatură astfel:**

```
.....
int v[100],i,n,j,ok,s=0;
int main()
{
    cout << "Lungimea vectorului n= "; cin>>n;
    for(i=0;i<=n-1;i++) cin>>v[i];
```

a)	b)	c)
<pre>for(i=0;i<=n-1;i++) { ok=0; for(j=2;j<=sqrt(v[i]);j++) if(v[i]%j==0) ok=0; if(ok==1) s=s+v[i]; } cout<<"Suma elementelor prime ="<<s;</pre>	<pre>for(i=0;i<=n-1;i++) { ok=1; for(j=2;j<=sqrt(v[i]);j++) if(v[i]%j==0) ok=0; if(ok==1) s=s+v[i]; } cout<<"Suma elementelor prime ="<<s;</pre>	<pre>for(i=0;i<=n-1;i++) { ok=1; for(j=2;j<=sqrt(v[i]);j++) if(v[i]/j==0) ok=0; if(ok==1) s=s+v[i]; } cout<<"Suma elementelor prime ="<<s;</pre>

- 15. Care este principiul de bază al căutării secvențiale?**

- Comparăm fiecare element cu valoarea căutată până găsim sau parcurgem tot șirul.
- Împărțim șirul în jumătăți și căutăm în mijloc.
- Căutăm elementul doar la început și la sfârșit.

- 16. Care este condiția pentru încheierea căutării secvențiale?**
- a. Se ajunge la mijlocul șirului.
 - b. Elementul căutat este găsit sau șirul este complet parcurs.
 - c. Șirul este ordonat crescător.
- 17. Căutarea secvențială funcționează pentru:**
- a. Șiruri ordonate și neordonate.
 - b. Numai șiruri ordonate.
 - c. Numai șiruri neordonate.
- 18. Cum determinăm poziția unui element găsit prin căutare secvențială?**
- a. Îl căutăm doar în partea dreaptă.
 - b. Memorăm indexul elementului în timpul parcurgerii șirului.
 - c. Este întotdeauna la mijloc.
- 19. Ce face căutarea binară diferită de căutarea secvențială?**
- a. Verifică elementele de la stânga la dreapta.
 - b. Utilizează mijlocul șirului pentru a reduce dimensiunea zonei de căutare.
 - c. Caută doar în șiruri neordonate.
- 20. Care este cerința principală pentru aplicarea căutării binare?**
- a. Șirul trebuie să fie ordonat.
 - b. Șirul trebuie să fie neordonat.
 - c. Toate elementele trebuie să fie pozitive.
- 21. Ce se întâmplă în căutarea binară dacă elementul căutat este mai mic decât mijlocul șirului?**
- a. Căutarea continuă în jumătatea din dreapta.
 - b. Căutarea continuă în jumătatea din stânga.
 - c. Elementul nu există.
- 22. În căutarea secvențială, ce facem dacă elementul curent nu este egal cu valoarea căutată?**
- a. Continuăm să verificăm următorul element.
 - b. Ne oprim din căutare.
 - c. Trecem la mijlocul șirului.
- 23. Cum se calculează mijlocul șirului în căutarea binară?**
- a. $\text{mijloc} = (\text{stanga} + \text{dreapta}) / 2$.
 - b. $\text{mijloc} = \text{stanga} + \text{dreapta}$.
 - c. $\text{mijloc} = \text{dreapta} - \text{stanga}$.
- 24. Care este avantajul principal al căutării binare?**
- a. Poate fi aplicată pe orice șir.
 - b. Reduce în mod semnificativ numărul de elemente verificate.
 - c. Funcționează doar pentru numere pozitive.

25. Care din afirmațiile următoare sunt adevărate referitoare la următoarea secvență de cod:

```
int n,x,i,eGasit,v[50];
cout<<"Dati numarul de elemente ale sirului : "; cin>>n;
cout<<"Introduceti elementele sirului"<<endl;
for(i=0; i<=n-1; i++) cin>>v[i];

cout<<"Dati numarul de cautat :"; cin>>x;

eGasit=0;
for(i=0; i<n; i++)
{
    if (v[i]==x)
    {
        eGasit=1;
        cout<<"Nr apartine sirului si se afla pe pozitia "<<i<<endl;
    }
}
if (eGasit==0) cout<<"Numarul nu apartine sirului!";
```

- Realizează o căutare secvențială a elementului **x** în vectorul **v**
- Realizează o căutare binară a elementului **x** în vectorul **v**
- Variabila **eGasit** este utilizată pentru a verifica dacă numărul căutat a fost găsit în șir.
- Numărul căutat este întotdeauna afișat împreună cu poziția sa, indiferent dacă există sau nu în șir.
- Șirul de valori pot fi introduse în orice ordine.

26. Care din afirmațiile următoare sunt adevărate referitoare la următoarea secvență de cod:

```
int v[100], n, i;
int x, eGasit=0, cstg, cdr, mijloc;

cout<<"n="; cin>>n;
for (i=0; i<=n-1; i++) cin>>v[i];
cout<<"Elementul cautat x="; cin>>x;
cstg=0; cdr=n-1;

while (cstg<=cdr && eGasit!=1)
{
    mijloc=(cstg+cdr)/2;

    if (v[mijloc]==x) eGasit=1;
    else
        if (v[mijloc]<x) cstg=mijloc+1;
        else cdr=mijloc-1;
}

if (eGasit==1) cout<<"Da, pe pozitia "<<mijloc;
else cout<<"Nu am gasit elementul";
```

- Codul folosește algoritmul de căutare binară pentru a găsi un element într-un șir.
- Codul folosește algoritmul de căutare secvențială pentru a găsi un element într-un șir.
- Șirul de elemente trebuie să fie ordonat pentru ca algoritmul să funcționeze corect.
- Șirul de elemente nu trebuie să fie ordonat pentru ca algoritmul să funcționeze corect
- Codul nu se oprește la prima apariție a elementului căutat ci va continua să caute duplicate.
- Programul va funcționa corect chiar dacă utilizatorul introduce un număr negativ pentru **n**
- La fiecare iterație, variabila **mijloc** reprezintă mijlocul intervalului în care se caută elementul.
- Dacă valoarea **a[mijloc]** este mai mică decât valoarea căutată **x** se continuă căutarea în jumătatea din dreapta a șirului.