

1. Ce reprezinta tipul referinta? Dati un exemplu unde se foloseste referinta.

Concept	Explicație	Exemplu
Referință ( & )	Un alias pentru o variabilă existentă	<code>int x = 10; int &amp;ref = x;</code>
Pointer ( * )	O variabilă care stochează adresa altui obiect	<code>int* p = &amp;x;</code>

2. Considerand ca aveti o clasa Test cu un membru A, de tip referinta si unul B de tip const, realizati constructorul clasei ce va initializa cei doi membri;

```
class Test {  
private:  
    int& A;        // Membru referință  
    const int B;   // Membru constant  
  
public:  
    // Constructor cu lista de inițializare  
    Test(int& ref, int val) : A(ref), B(val) {}  
}
```

3. Unde este utilizat operatorul ->\* in standardul C++ ? Exemplificati.

Folosit pentru a accesa un pointer la membru (variabilă sau funcție) dintr-o clasă, printr-un pointer la obiect

```
class Test { public: int x = 10; };  
int main() {  
    Test obj, *ptr = &obj;  
    int Test::*p = &Test::x;  
    ptr->*p = 20;  
}
```

4. Dati un exemplu de functie „friend”;

```
friend istream& operator>>(istream& c, Persoana& p) {  
    cout << "Introdu numele: ";  
    c >> p.nume;  
    cout << "Introdu varsta: ";  
    c >> p.varsta;  
    return c;  
}
```

## 5. Ce este o clasa abstracta? Ce proprietati intalnim la o astfel de clasa?

**Clasa abstractă** nu poate fi instanțiată direct.

### Proprietăți:

- **Metode virtuale pure** :O metodă virtuală pură este o funcție membru definită în clasa abstractă, dar fără implementare, iar în locul implementării, se folosește sintaxa `=0`

```
virtual void metoda() = 0;
```

- **Nu poate fi instanțiată**: Nu poți crea obiecte direct dintr-o clasă abstractă
- **Poate avea membri concreți (cu implementare)**  
O clasă abstractă poate conține metode și membri de date care au implementare (nu trebuie să fie toate virtuale sau pure).
- **Clasa derivată trebuie să implementeze toate metodele virtuale pure**  
O clasă derivată va trebui să furnizeze implementări pentru toate metodele virtuale pure din clasa abstractă.

## 6. Cum se implementează în C++ polimorfismul ad-hoc? Dati un exemplu de polimorfism ad-hoc. Dati un exemplu de clasa template.

**Polimorfismul ad-hoc** se referă la utilizarea aceluiași nume de funcție sau operator pentru a face lucruri diferite în funcție de tipurile de argumente. Acesta este realizat prin **suprascrierea funcțiilor** (overloading) și **suprascrierea operatorilor**.

```

class Exemplu {
public:
    // Suprascriere funcție pentru tipul int
    void afiseaza(int x) {
        cout << "Valoare int: " << x << endl;
    }

    // Suprascriere funcție pentru tipul double
    void afiseaza(double x) {
        cout << "Valoare double: " << x << endl;
    }
};

int main() {
    Exemplu obj;

    obj.afiseaza(5);        // Apel funcție pentru int
    obj.afiseaza(3.14);     // Apel funcție pentru double
}

```

```

#include <iostream>
using namespace std;

// Funcție template pentru a returna valoarea unui argument
template <typename T>
T returneaza(T valoare) {
    return valoare;
}



int main() {
    int x = 5;
    double y = 3.14;

    // Apel pentru tipul int
    cout << "Valoarea pentru x: " << returneaza(x) << endl;

    // Apel pentru tipul double
    cout << "Valoarea pentru y: " << returneaza(y) << endl;

    return 0;
}

```

 Copy  Edit

**8. Analizand urmatorul scenariu va rog sa specificati ( intr-o viziune POO ) care din entitatite luate in discutie ( marcate) pot fi clase, respectiv instante si definiti relatia dintre ele.**

**Afara se jucau patru copii. Ion si Vlad jucau fotbal iar Mirela si Ioana se juca de-a baba-carba.**

**Clasă **Copil**:**

- Reprezintă un copil cu attribute comune (ex. **nume**).
- Instantele: Ion, Vlad, Mirela, Ioana.

**Clasă **Joc**:**

- Reprezintă un joc cu attribute comune (ex. **tip\_joc**).
- Instantele: Fotbal, Baba-Carba.

**Relația dintre ele:**

- Un **Copil** poate participa la un **Joc**.
- Fiecare **Copil** poate juca un **Joc** diferit sau mai multe jocuri.