Instructiuni decizionale.

Instructiuni iterative.

Instructiuni decizionale

- Instructiunea **`if`**
- Instructiunea **`if-else`**
- Instructiunile **`if` imbricate**
- Instructiunea **`swith`**
- Operatorul conditional **`?:`**

Instructiuni iterative

- Instructiunea **`for`**
- Instructiunea **`while`**
- Instructiunea **`do-while`**
- Instructiunile **`break`** si **`continue`**
- Instructiuni iterative (bucle) infinite
- Instructiuni iterative imbricate

Instructiunea decizionala **if**

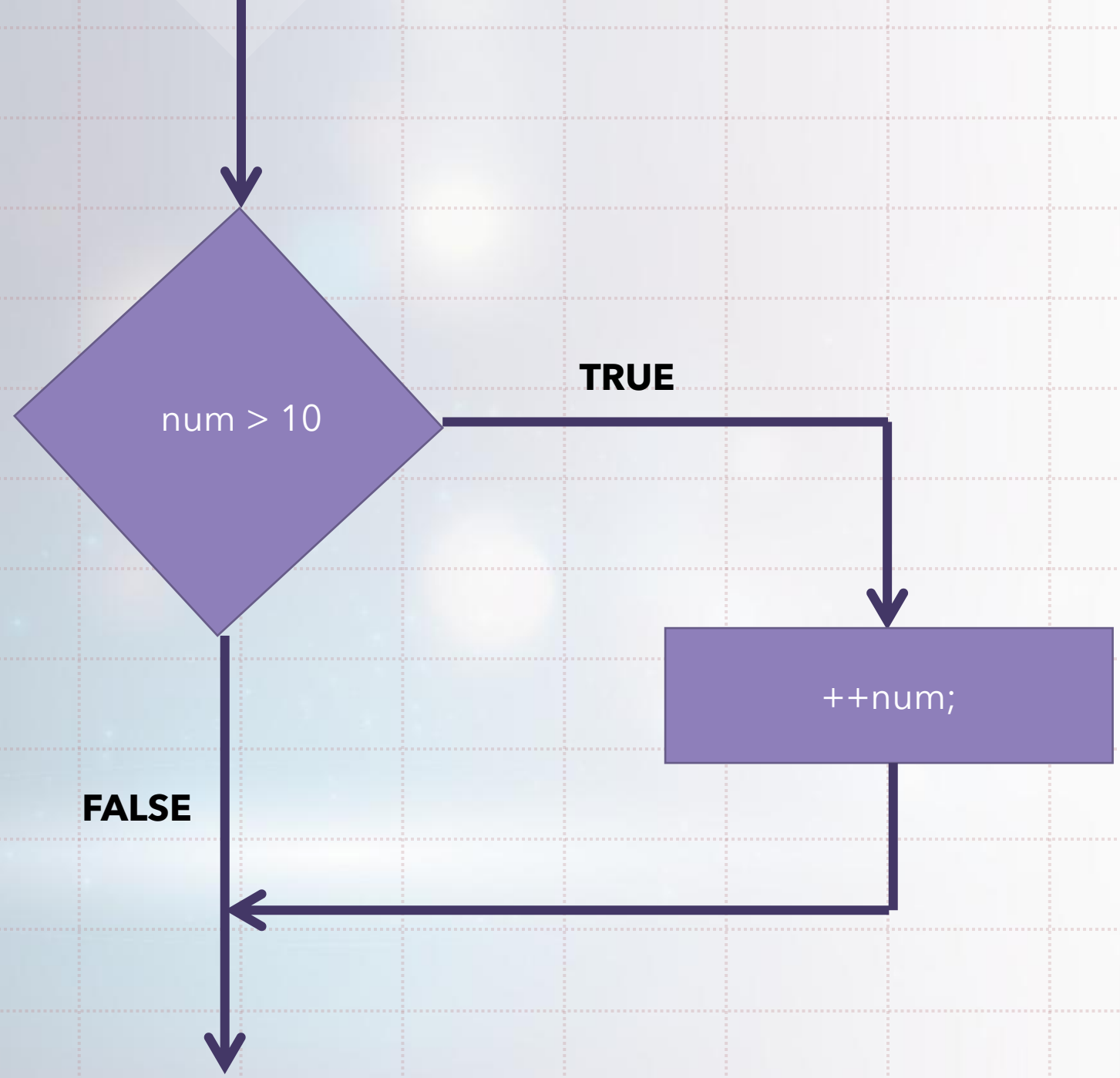
Sintaxa:

```
if (expr_booleana)
        instructiune;
```

- Daca expresia **`expr_booleana`** este adevarata atunci se executa instructiunea
- Daca expresia **`expr_booleana`** este falsa nu se executa instructiunea

- Avem trei modalitati echivalente de a scrie:
  - `if(expr_booleana)`
  - `if(expr_booleana==true)`
  - `if(expr_booleana==1)`

```
if (num > 10)
    ++num;
```



Instructiunea decizionala **if**

Exemple:

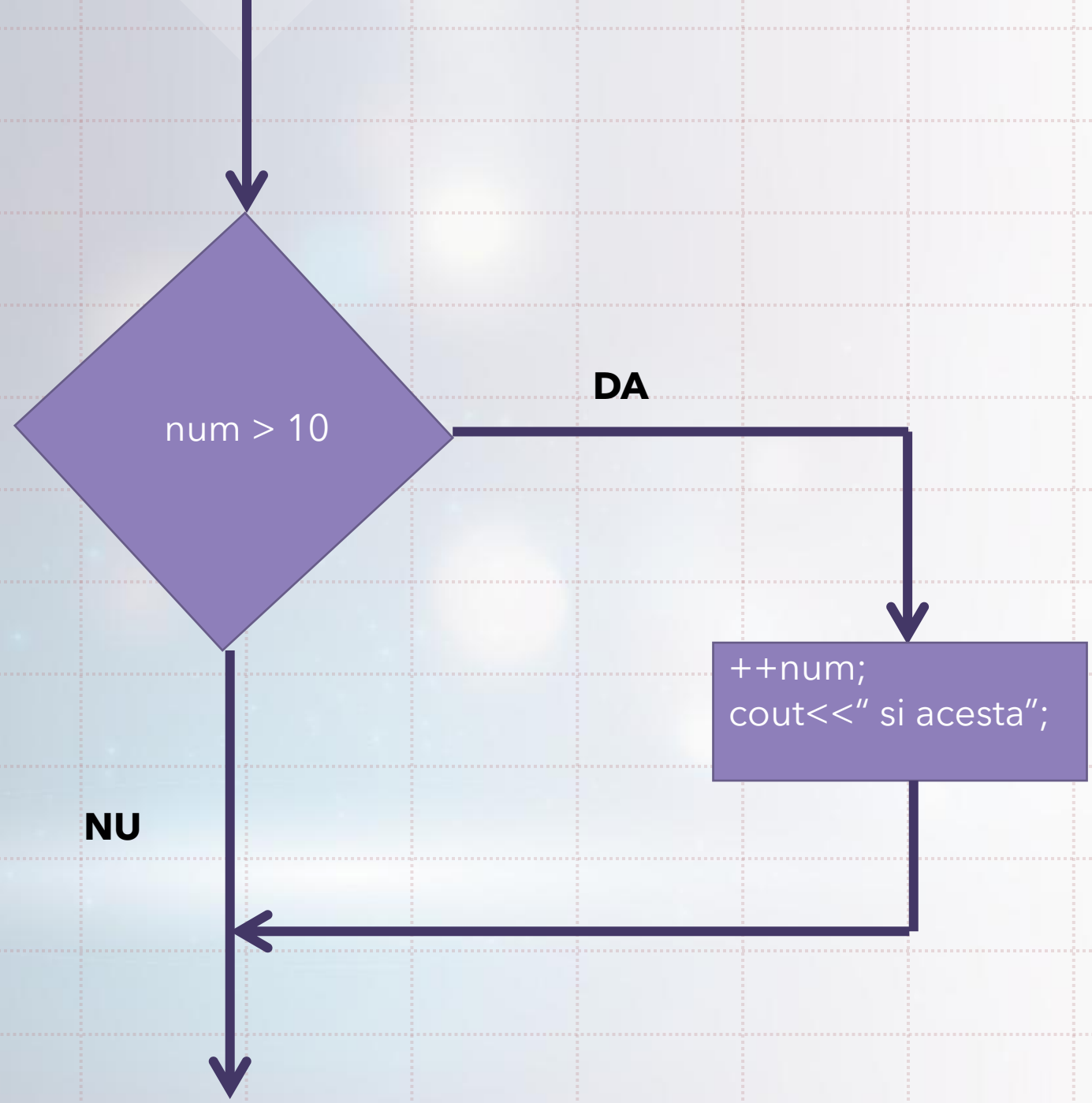
```
if ( selectie == ‘A’)
       cout<<“Ati selectat A”<<endl;

if ( num > 10)
       cout<<“Num este mai mare decat 10”<<endl;

if ( varsta<5 && talieMica)       //daca ambele expresii sunt adevarate
       cumpar=true;
```

Atentie!
In cazul in care instructiunea conditionala e compusa din mai multe linii trebuie sa folosim **acolade**.

```
if ( num > 10){
        ++num;
        cout<<“ si acesta”;
}
```



Instructiunea decizionala **if**

```
if (expr_booleana)
        {
                // declarari de variabile
                Instructiune1;
                Instructiune2;
                ……
        }
```

In cadrul instructiunii conditionale putem avea mai multe instructiuni si declarari de variabile.

Acest variabile sunt vizibile doar in cadrul instructiunii unde au fost declarate.

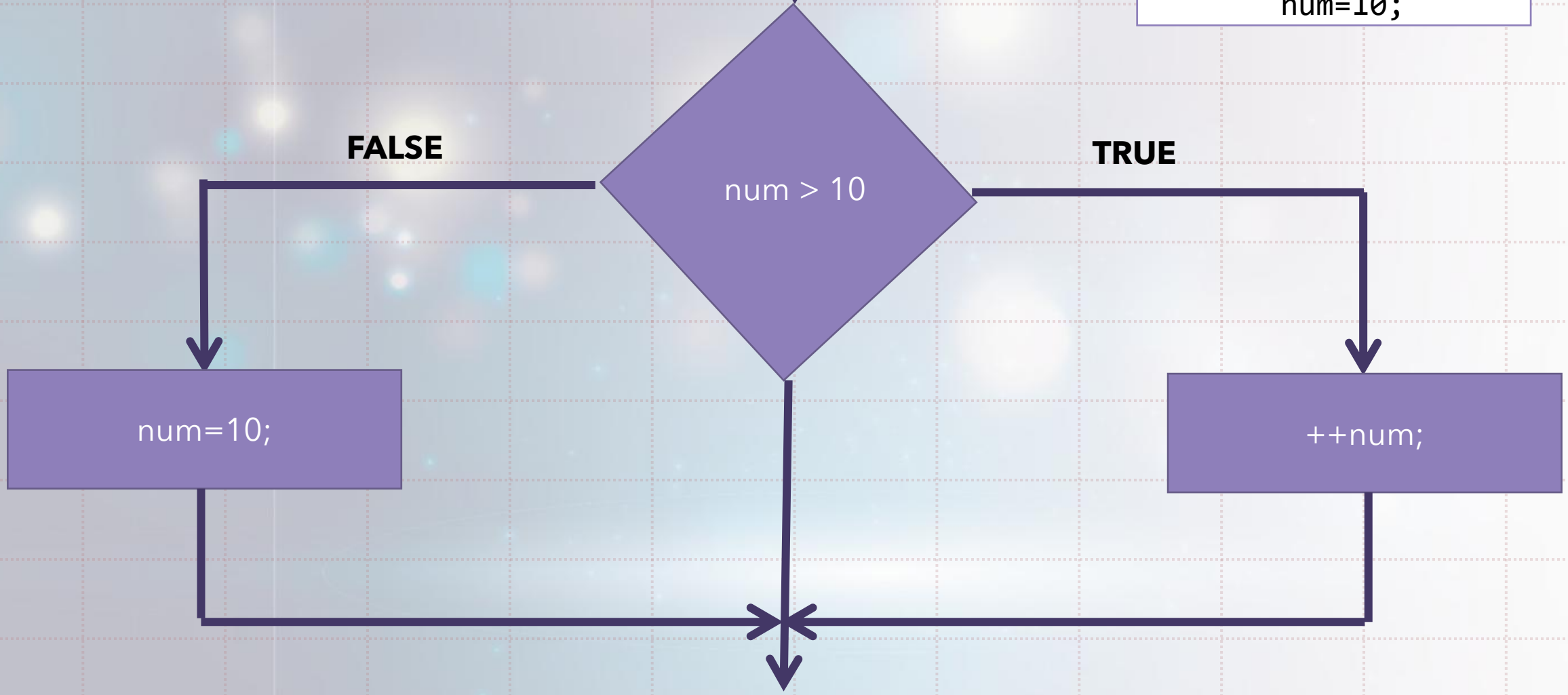
Vezi programul “InstructiuneIf”!

Instructiunea decizionala **if else**

```
if (expr_booleana)
        instructiune1;
else
        instructiune2;
```

- Daca **expr_booleana** este adevarata atunci se executa **instructiune1**
- Daca **expr_booleana** este falsa atunci se executa **instructiune2**

```
if ( num > 10)
        ++num;
else
        num=10;
```



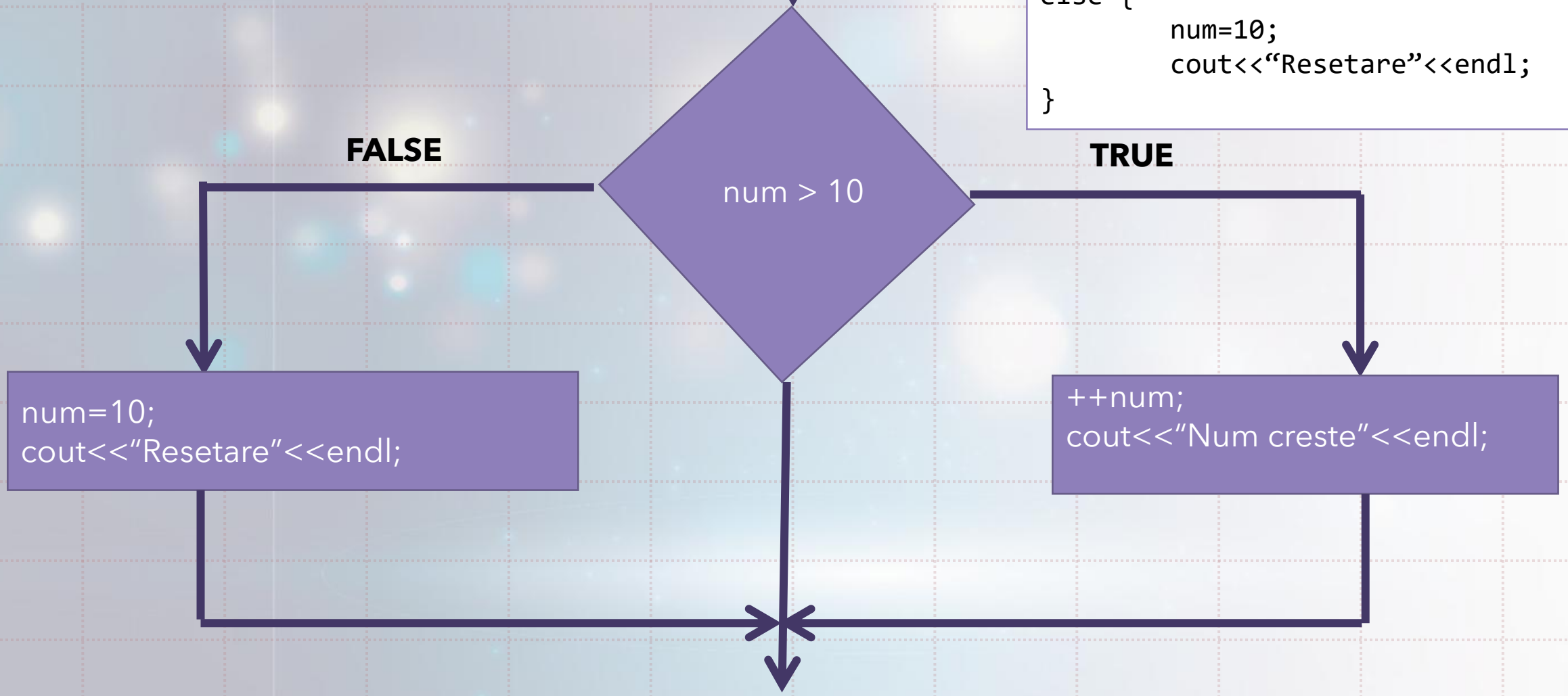
Instructiunea decizionala **if-else**

Exemple:

```
........
if (num > 10)
   cout << “num este mai mare decat 10” << endl;
else
   cout << “num nu este mai mare decat 10” << endl;

.....
if (varsta < 5 && talieMica)        //daca ambele expresii sunt adevarate
       cumpar = true;
else
       cout << “Nu este un caine potrivit  pentru mine!” << endl;
```

```
if (num > 10) {
        ++num;
        cout<<“Num creste”<<endl;
}
else {
        num=10;
        cout<<“Resetare”<<endl;
}
```



Vezi programul “InstructiuneIfElse”!

Instructiunea decizionala **if-else if**

Exemplu:

```
if (nota >= 90)
        cout << “Foarte bine”;
else if (nota >= 70)
        cout << “Bine”;
else if (nota >= 50)
        cout << “Satisfacator”;
else
cout << “Insuficient”;
cout << “Gata!”;
```

Instructiunile if imbricate

```
if (expr1)
        if (expr2)
                instructiune1;
        else
                instructiune2;
```

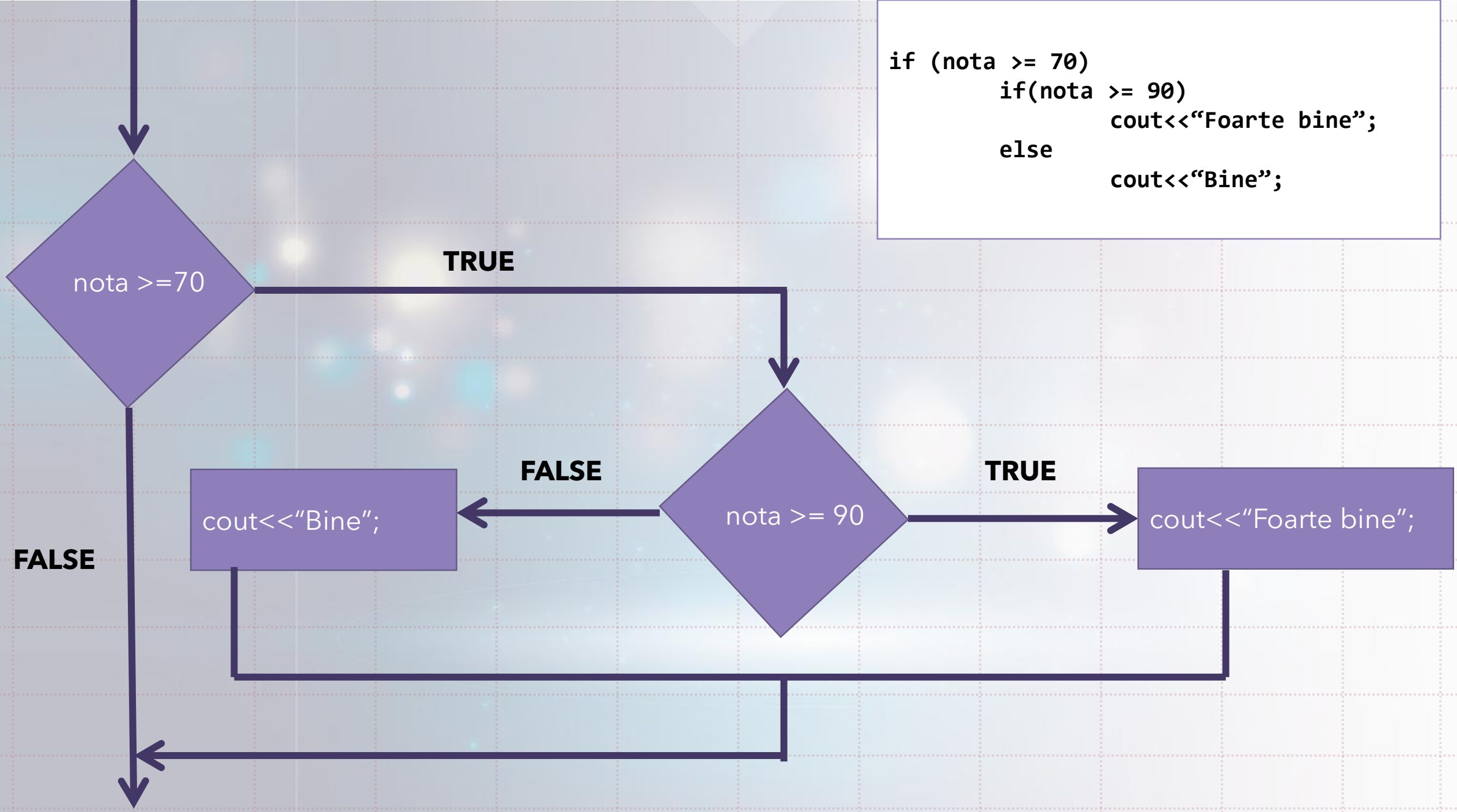
- Instructiunea **if** poate sa apara in interiorul altei instructiuni **if**. In acest caz se spune ca o instructiune **if** este imbricata in interiorul altei instructiuni **if**.
- Se folosesc pentru a testa multiple conditii

Instructiunile if imbricate

Exemplu:

```
if (nota >= 70)
        if (nota >= 90)
                cout << “Foarte bine”;
        else
                cout << “Bine”;
else
        cout << “Nu este bine!”;
```

```
if (nota >= 70)
        if(nota >= 90)
                cout<<“Foarte bine”;
        else
                cout<<“Bine”;
```



Instructiunile `if` imbricate

Exemplu:

```
if (scor_Maria != scor_Ioana){
        if (scor_Maria > scor_Ioana) {
                cout << “Maria a castigat” << endl;
        } else {
                cout<< “Ioana a castigat” << endl;
           }
} else {
        cout << “Este egalitate!” << endl;
}
```

Vezi programele “InstructiunileIfImbricate”, “InstructiunileIfImbricate2”!

Instructiunea **switch**

```
switch (expr_control){
        case expr_1: instructiune_1; break;
        case expr_2: instructiune_2; break;
        case expr_3: instructiune_3; break;
        …
        case expr_n: instructiune_n; break;
        default: instructiune_implicita;
}
```

- Instrucțiunea switch este o **alternativă mai clară și mai ordonată** la o succesiune de instrucțiuni `if - else if - else`. Ea **verifică valoarea unei expresii** și **execută blocul de cod corespunzător** etichetei (sau "cazului") potrivite.

- `expr_control` trebuie sa fie o o variabila de tip **intreg**, **caracter** sau de tip **enum**. In functie de *`expr_control`*, se va executa unul din cazurile *`case expr_1, expr_2, expr_3,…, expr_n`*. Dacă niciuna din aceste valori *`expr_i`* nu este egală cu valoarea variabilei *`expr_control`*, se vor executa instructiunile din ramura *`default`* (dacă aceasta există).

- Prezenta instructiunii *`break`* asigură executarea unui singur caz (`case`); în lipsa ei, executându-se toate cazurile (inclusiv *`default`*), începând cu cel pentru care *`expr_i`* este egal cu *`expr_control`*.

Instructiunea **switch**

Exemplu:

```
char selectie;
cout<<“Introduceti un caracter: “;
cin>>selectie;
switch (selectie){
      case ‘1’: cout<<“Ati selectat 1”;
            break;
      case ‘2’: cout<<“Ati selectat 2”;
            break;
      case ‘3’:
      case ‘4’: cout<<“Ati selectat 3 sau 4”;
            break;
      default: cout<<“Nu ati selectat 1,2,3 sau 4!”;
}
```

- *selectie* este un caracter (care poate fi gandit si ca un intreg) deoarece fiecarui caracter ii corespunde un intreg (intre 0 si 127)

Instructiunea **switch**

Exemplu:

```
switch (selectie){
        case ‘1’: cout<<“Ati selectat 1 ”;
        case ‘2’: cout<<“Ati selectat 2 ”;
        case ‘3’: cout<<“Ati selectat 3 ”;
        case ‘4’: cout<<“Ati selectat 4 ”;
                  break;
        default: cout<<“Nu ati selectat 1,2,3 sau 4!”;
}
```

- Daca selectie=‘2’, ce se afiseaza?

“Ati selectat 2 Ati selectat 3 Ati selectat 4”

Instructiunea **switch**

Exemplu:

```
enum Culoare{rosu, verde, albastru};

Culoare culoare_preferata {verde};

switch (culoare_preferata){
        case rosu: cout<<“Culoarea preferata este rosu”; break;
        case verde: cout<<“Culoarea preferata este verde”; break;
        case albastru: cout<<“Culoarea preferata este albastru”;
break;

        default: cout<<“Niciodata nu se executa”;
}
```

Instructiunea **`switch`**

- **Remember!**

- **`Expr_control`** trebuie sa fie de tip intreg, caracter sau enum!
- Odata ce expresia corespunzatoare unui anumit caz coincide cu **`expr_control`** atunci toate celelalte cazuri se executa **pana** se intalneste instructiunea **`break`**!
- Este recomandat sa folosim **`break`** dupa fiecare caz!
- Este recomandat sa avem si cazul **`default!`**

Vezi programele “InstructiuneSwitch” si “InstructiuneSwitch2”!

Operatorul conditional ? :

```
(expr_cond) ? expr1 : expr2
```

- Este un operator ternar, cei trei operanzi fiind **`expr_cond`**, **`expr1`** si **`expr2`**

- **`expr_cond`** este o expresie booleana:
  - ❖ Daca **`expr_cond == true`**, atunci se returneaza valoarea lui **`expr1`**
  - ❖ Daca **`expr_cond == false`**, atunci se returneaza valoarea lui **`expr2`**

- Este similar cu instructiunea **`if-else`**
- Este foarte folositor si sintaxa sa este pe o singura linie

Operatorul conditional ?:

- Exemplu

```
int a {10}, b {20};
int scor {92};
int rezultat { };

rezultat = (a>b) ? a : b
rezultat = (a<b) ?(b-a) : (a-b);
rezultat = (b!=0) ? (a/b) : 0;
cout<<((scor>90) ? “Excelent” : “Good”);
```

rezultat=b=20

rezultat=b-a=10

rezultat=a/b=0

Excelent

Vezi programul “OperatorulConditional”!

Instructiuni ciclice/de ciclare /iterative/ repetitive

- Permit executia unei secvente de cod in mod repetat
- Numărul de repetiţii poate fi fix, stabilit iniţial, sau variabil, depinzând de rezultatul acţiunii repetate.
- Cand putem folosi instructiunile de ciclare?

  - Cand vrem ca o secventa de cod sa se repete de un numar precis de ori
  - Pentru fiecare element al unei colectii (tablou)
  - Cat timp o anumita conditie ramane adevarata
  - Pana cand o anumita conditie devine falsa
  - Pana cand ajungem la finalul unui input
  - Si alte cazuri.

Instructiuni de ciclare (iterative/ repetitive)

- Instructiunea **for**
  --- iteratia se face de un numar precis de ori
- Instructiunea **ranged-based for**
  --- o iteratie pentru fiecare element dintr-o colectie
- Instructiunea **while**
  --- iteratia se face cat timp o anumita conditie ramane adevarata;
  --- se opreste cand conditia devine falsa
  --- se verifica conditia la inceputul fiecarei iteratii
- Instructiunea **do-while**
  --- iteratia se face cat timp o anumita conditie ramane adevarata;
  --- se opreste cand conditia devine falsa
  --- se verifica conditia la finalul fiecarei iteratii

Instructiunea for

- Sintaxa:

```
for  ( expresie_initiala; expresie_test; expresie_repetitiva ) {
     instructiune_corp
   }
```

- Expresiile din cadrul instructiunii **for** sunt separate prin "**;**" ca sa sugereze faptul ca ele sunt optionale

- Instrucţiunea **for** furnizează o scriere compactă a etapelor standard de execuţie a unui proces iterativ:
  - ✓ iniţializarea variabilelor înaintea startului,
  - ✓ evaluarea testului de ciclare,
  - ✓ execuţia acţiunilor din corpul ciclului,
  - ✓ actualizarea variabilelor implicate în **expresia_test**,
  - ✓ reluarea testării

Instructiunea for

- **Exemplu:**

```
int i {0};
for (i = 1; i<=5; i++)
//acelasi lucru se intampla daca scriem for (i = 1; i<=5; ++i)
       cout << i << endl;
```

1
2
3
4
5

Instructiunea for

- **Exemplu:**

```
for (int i {1}; i<=5; i++)
        cout << i << endl;

//stilul de initializare
```

1
2
3
4
5

- **Exemplu:**

```
for (int i =1; i<=5; i++)
        cout << i << endl;

//stilul de atribuire
```

- Variabila `i` este vizibila doar in interiorul instructiunii `for`

Instructiunea for

- **Exemplu: Afisati numele pare dintre 1 si 10**

```
for (int i {1}; i<=10; i++) {
        if (i %2 ==0)
                cout << i << endl;
}
```

```
2
4
6
8
10
```

- **Exemplu: Afisati elementele unui tablou.**

```
int scor [ ]  {100,90,87};

for (int i {0}; i<3; i++) {
        cout << scor [i] << endl;
}
```

```
int scor [ ]  {100,90,87};

for (int i {0}; i<=2; i++) {
        cout << scor [i] << endl;
}
```

```
100
90
87
```

Instructiunea for

- **Operatorul ,**

```
for (int i {1}, j {5} ; i<=5 ;  i++, j++) {
        cout << i << " * "<<j<< " : "<< (i*j)<<endl;
}
```

- Operatorul "," ne permite sa speraram mai multe expresii si sa executam toate expresiile

```
1 * 5 : 5
2 * 6 : 12
3 * 7 : 21
4 * 8 : 32
5 * 9 : 45
```

Instructiunea for

Alte detalii:

- Sintaxa pentru instructiunea `for` standard este foarte clara si concisa
- Cum toate expresiile din cadrul instructiunii `for` sunt optionale, este posibil sa nu avem nici initializare, nici testare, nici incrementare

```
for(; ;)
        cout<<“Iteratie infinita”<<endl;
```

Iteratie infinita
Iteratie infinita
Iteratie infinita
….
….

Vezi programul “InstructiuneaFor”!

Instructiunea range-based for

- Introdusa in C++11

```
for (tipul_var  nume_var: colectie){
        bloc_instructiuni;                    //se poate folosi nume_var
}
```

Exemplu:

```
int note [ ] {8, 10, 9};

for (int nota : note)
        cout<<nota<<endl;
```

```
8
10
9
```

Instructiunea range-based for

- Avem si optiunea sa nu mai precizam tipul variabilei si atunci in loc de “**tipul_var**” vom scrie “**auto**”

Exemplu:

```
int note [ ] {8, 10, 9};

for (auto nota : note)
        cout<<nota<<endl;
```

```
8
10
9
```

Vezi programul “RangedBasedFor”!

Instructiunea `while`

```
while (expresie_booleana) {
        bloc_instructiuni;
}
```

- Este un exemplu de iteratie in care testarea se face prima data
- **`expresie_booleana`** are fie valoarea **`true`**, fie **`false`**
- Daca testarea cade (**`expresie_booleana==false`**) atunci nu se mai executa instructiunile din interiorul iteratiei

Exemplu:

```
int i {1};
while (i <=5){
        cout<<i<<endl;
        ++i; // este important ca i sa fie incrementat pentru ca altfel am avea
o
             //iteratie infinita
}
```

Instructiunea while

- De multe ori se foloseste instructiunea while pentru a verifica un text introdus de programator (**input validation**)

Exemplu:

```
int numar{};
cout << "Introduceti un numar mai mic decat 100: ";
cin >> numar;

while (numar >= 100) { // sau echivalent, !(numar < 100)
       cout << "Introduceti un numar mai mic decat 100: ";
       cin >> numar;
       }
cout << "Multumesc! " << endl;
```

Instructiunea while

- **Input validation**

Exemplu:

```
int numar{};
cout << "Introduceti un numar cuprins intre 1 si 5 (strict):  ";
cin >> numar;

while (numar<=1 || numar>=5) {
        cout << "Introduceti un numar intre 1 si 5 (strict): ";
        cin >> numar;
        }
cout << "Multumesc! " << endl;
```

Instructiunea while

- **Input validation – cu ajutorul unei variabile de tip bool**

Exemplu:

```
bool ok{ false };
int numar{};
while (!ok) { //ok==false
        cout << "Introduceti un numar intre 1 si 5 (strict): ";
        cin >> numar;
        if (numar <= 1 || numar >= 5)
                cout << "Numarul este in afara intervalului, mai incercati o data! " << endl;
        else {
                cout<< "Multumesc! " << endl;
                ok = true;
                }
        }
```

- Vezi programul “InstructiuneaWhile”!

Instructiunea do-while

```
do{
        bloc_instructiuni;
} while (expresie_booleana);
```

- Se executa **bloc_instructiuni** cat timp **expresie_booleana==true**
- Expresia conditionala se verifica la sfarsitul fiecarei iteratii, deci **bloc_instructiuni** se va executa macar o data.

Exemplu: input validation

```
int numar{};
do {
        cout << "Introduceti un numar intre 1 si 5 (strict): ";
        cin >> numar;
} while (numar <= 1 || numar >= 5);

cout << "Multumesc!";
```

- Vezi programul “InstructiuneaDoWhile”!

Instructiunile continue si break

- **continue**
  - Nici o instructiune care este in continuare in corpul buclei nu se mai executa
  - Se trece direct la inceputul buclei pentru iteratia urmatoare

- **break**
  - Nici o instructiune care este in continuare in corpul buclei nu se mai executa
  - Bucla se termina imediat

- Exemplu:

```
int valori[] = { 1,2,-1,3,-1,-99,7,8,10 };

for (auto val : valori) {
        if (val == -99)
                break;
        else if (val == -1)
                continue;
        else
                cout << val << endl;
}
```

```
1
2
3
```

Instructiunile repetitive infinite

- Sunt instructiunile a caror expresie conditionala este intotdeauna evaluata ca fiind adevarata
- De obicei este o eroare a programatorului
- Uneori programatorii folosesc bucle infinite si instructiunea break pentru a iesi din bucla
- Uneori buclele infinite sunt exact ceea ce avem nevoie: sistemul de operare, event-driven programming (programe in care ceea ce se intampla in cadrul lor este determinat de evenimente precum actiuni ale user-ului (click-ul mouse-ului, apasarea anumitor taste, mesaje s.a.) ; se inchid atunci cand se inchide calculatorul)

Instructiunile repetitive infinite

Exemple:

```
for (; ;)
        cout<< “Acesta text va aparea la infinit!”<<endl;


while(true) //while(9<10)
        cout<< “Acesta text va aparea la infinit!”<<endl;



do {
        cout<< “Acesta text va aparea la infinit!”<<endl;
} while (true);
```

Instructiunile repetitive infinite

Exemplu:

```
while(true){
        char din_nou;
        cout<< “Doriti o noua iteratie? (D/N): ”;
        cin>>din_nou;

        if (din_nou==‘n’ || din_nou==‘N’)
        break;
}
```

Instructiunile repetitive imbricate

- Instructiunea **repetitiva** poate sa apara in interiorul altor instructiuni **repetitive**.
- Sunt foarte folositoare in multi-dimensiuni (de exemplu la tablouri bidimensionale)
- **Instructiune repetitiva exterioara** vs. **instructiune repetitiva interioara**

Instructiunile repetitive imbricate

- Exemplu:

```
for (int val_exterioara{ 1 }; val_exterioara <= 2; val_exterioara++)
        for (int val_interioara{ 1 }; val_interioara <= 3; val_interioara++)
                cout << val_exterioara << ", " << val_interioara << endl;
```

```
1, 1
1, 2
1, 3
2, 1
2, 2
2, 3
```

Vezi programul “InstructiuniRepetitiveImbricate”!

https://test-master.space