PROGRAMARE II
SUBIECT EXAMEN [MODEL]

CITIȚI CU ATENȚIE ENUNȚUL PROBLEMEI ȘI BAREMUL DE NOTARE

Să se implementeze o clasă **Publicatie** cu următorii membri:

- **Titlu** (de tip **string**)
- **NrPagini** (de tip **unsigned int**)

Pentru clasa Publicatie se vor implementa următoarele metode:

- Constructor fără argumente care inițializează datele de tip string cu un șir vid,

iar cele de tip intreg cu 0;

- Constructor de inițializare (cu listă de parametri) care va inițializa membrii clasei cu valorile parametrilor;
- Destructor ce realizează dealocarea tuturor spațiilor de memorie alocate dinamic;
- O metodă care va afișa informațiile publicației.

CITIȚI CU ATENȚIE ENUNȚUL PROBLEMEI ȘI BAREMUL DE NOTARE

Să se implementeze clasa **Revista**:

- **Editura** (de tip **string**)
- **Tiraj** (de tip **unsigned int**)

Pentru clasa Revista se vor implementa următoarele metode:

- Constructor fără argumente care inițializează datele de tip string cu șir vid, iar cele de tip intreg cu 0
- Constructor de inițializare (cu listă de parametri) care va inițializa membrii clasei cu valorile parametrilor și apelează un constructor din clasa de bază pentru inițializarea membrilor moșteniți
- Destructor ce realizează dealocarea tuturor spațiilor de memorie alocate dinamic
- Supraîncărcarea unui operator care returnează rezultatul concatenării câmpurilor titlu și editura ale unui obiect de tip Revista
- O metodă de afișare a informațiilor revistei (ce are aceeași declarație ca metoda echivalentă din clasa Publicatie)

CITIȚI CU ATENȚIE ENUNȚUL PROBLEMEI ȘI BAREMUL DE NOTARE

Cerințe:

- Creați un obiect de tip Publicatie și unul de tip Revista și testați toate metodele implementate din clasele corespunzătoare
- Creați un tablou de pointeri la clasa Publicatie ce conține:
  - 3 obiecte de tip Publicație
  - 3 obiecte de tip Revista
  - Testați metoda de afișare utilizând numai obiectele din acest tablou:
    - se va avea în vedere polimorfismul (aplicația trebuie să asigure faptul că pentru obiectele de tip Publicatie se va apela metoda din clasa Publicatie, iar pentru obiectele de tip Revista se va apela metoda din clasa Revista)
- Calculați și afișați tirajul total al celor 3 reviste.
- Sortați descendent după numărul de pagini obiectele din tablou (bonus 0.5p pentru utilizarea funcției sort)

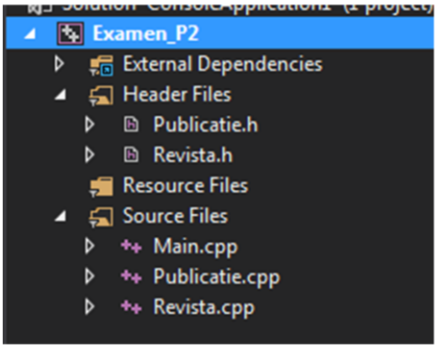
CITIȚI CU ATENȚIE ENUNȚUL PROBLEMEI ȘI BAREMUL DE NOTARE

Observații:

- Se va utiliza specificatorul de acces "public" doar pentru metodele claselor (nerespectarea acestei prevederi atrage dupa sine înjumătățirea punctajului acordat).
- Se acorda bonus 0.5p pentru evitarea utilizării pointerului "this" (acolo unde se poate)
- Fiecare clasa va avea headerul + sursele proprii, iar definirea metodelor se va realiza în fisierul sursă (.cpp)
- Clasele se pot completa, la nevoie, cu alte metode ajutătoare
- Numele membrilor claselor poate fi modificat dacă se dorește utilizarea unor convenții de notare

PENTRU A PROMOVA EXAMENUL

CREAREA PROIECTULUI



- Fiecare clasa va avea headerul + sursele proprii, iar definirea metodelor se va realiza în fisierul sursă (.cpp)

HEADERUL PUBLICATIE.H

Publicatie

```
#pragma once

class Publicatie
{
    string Titlu;
    unsigned int NrPagini;
public:
    Publicatie();
    //Publicatie(Publicatie &p);
    Publicatie(string t, unsigned int np);
    ~Publicatie();
    virtual void afisare();
    string getTitlu();
    unsigned int getNrPagini();
    bool operator<(const Publicatie &b);
};
```

Să se implementeze o clasă **Publicatie** cu următorii membri:

- **Titlu** (de tip **string**)
- **NrPagini** (de tip **unsigned int**)

Pentru clasa Publicatie se vor implementa următoarele metode:

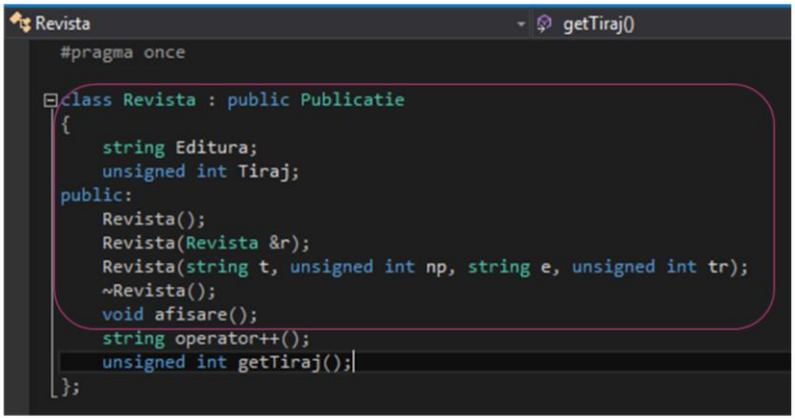
- Constructor fără argumente care inițializează datele de tip string cu un șir vid,

iar cele de tip intreg cu 0;

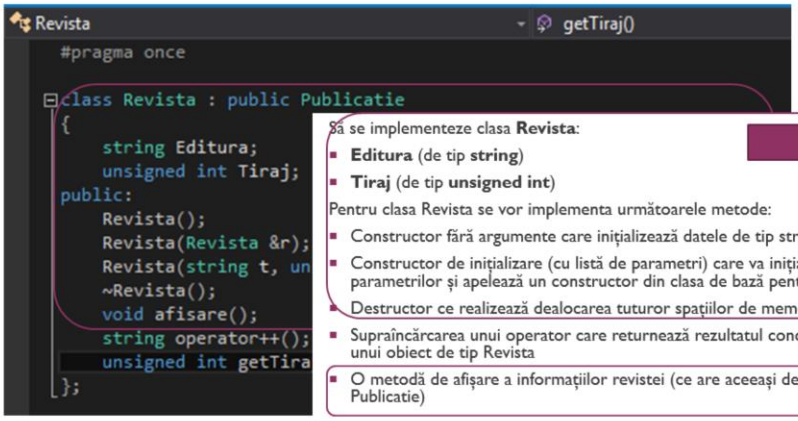
- Constructor de inițializare (cu listă de parametri) care va inițializa membrii clasei cu valorile parametrilor;
- Destructor ce realizează dealocarea tuturor spațiilor de memorie alocate dinamic;
- O metodă care va afișa informațiile publicației.

1p +1.75p = 2.75p

HEADERUL REVISTA.H



HEADERUL REVISTA.H



Să se implementeze clasa **Revista**:

2.75p + 2.25p = 5p

- **Editura** (de tip **string**)
- **Tiraj** (de tip **unsigned int**)

Pentru clasa Revista se vor implementa următoarele metode:

- Constructor fără argumente care inițializează datele de tip string cu șir vid, iar cele de tip intreg cu 0
- Constructor de inițializare (cu listă de parametri) care va inițializa membrii clasei cu valorile parametrilor și apelează un constructor din clasa de bază pentru inițializarea membrilor moșteniți
- Destructor ce realizează dealocarea tuturor spațiilor de memorie alocate dinamic
- Supraîncărcarea unui operator care returnează rezultatul concatenării câmpurilor titlu și editura ale unui obiect de tip Revista
- O metodă de afișare a informațiilor revistei (ce are aceeași declarație ca metoda echivalentă din clasa Publicatie)

PUBLICATIE.CPP [PART 1]

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

#include "Publicatie.h"

Publicatie::Publicatie()
{
    Titlu = "";
    NrPagini = 0;
}

Publicatie::Publicatie(string t, unsigned int np)
{
    Titlu = t;
    NrPagini = np;
}

Publicatie::~Publicatie()
{
}

void Publicatie::afisare()
{
    cout << "\nTitlul: " << Titlu << endl;
    cout << "Nr.pagini: " << NrPagini << endl;
}
```

MAIN.CPP [PART I]

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;
#include "Publicatie.h"
#include "Revista.h"

unsigned int tirajTotal(Publicatie** vp, int n){ ... }
void sortareClasica(Publicatie** &vp, int n){ ... }
void sortareSort(vector<Publicatie> &svp){ ... }

int main()
{
    Publicatie p1, p2("Publicatia", 3);
    Revista r1, r2("Publicatia_p", 33, "Revista_p", 500);
    p1.afisare();
    p2.afisare();
    r1.afisare();
    r2.afisare();
```

Creați un obiect de tip Publicatie și unul de tip Revista și testați toate metodele implementate din clasele corespunzătoare

5p+1.25p=6.25p

PENTRU NOTA > 6

SUPRAÎNCĂRCAREA OPERATORULUI DE CONCATENARE DIN CLASA REVISTA

- Supraîncărcarea unui operator care returnează rezultatul concatenării câmpurilor titlu și editura ale unui obiect de tip Revista
- Se va utiliza specificatorul de acces "public" doar pentru metodele claselor (nerespectarea acestei prevederi atrage dupa sine înjumătățirea punctajului acordat).
- Se acorda bonus 0.5p pentru evitarea utilizării pointerului "this" (acolo unde se poate)

```
string Publicatie::getTitlu()
{
    return Titlu;
}
```

```
string Revista::operator++()
{
    return getTitlu() + " " + Editura;
}
```

```
Publicatie p1, p2("Publicatia", 3);
Revista r1, r2("Publicatia_p", 33, "Revista_p", 500);
p1.afisare();
p2.afisare();
r1.afisare();
r2.afisare();

string rezConcat = ++r2;
cout << "\n\nSupraincarcare operator concat: " << rezConcat << endl;
```

6.25p + 0.75p = 7p

TABLOUL DE POINTERI LA CLASA PUBLICAȚIE

```
int n = 6;
Publicatie** vp = new Publicatie*[n];
vp[0] = new Publicatie("Publicatia1", 3);
vp[1] = new Publicatie("Publicatia2", 30);
vp[2] = new Publicatie("Publicatia3", 300);
vp[3] = new Revista("Publicatia4", 1, "Revista1", 10);
vp[4] = new Revista("Publicatia5", 2, "Revista2", 100);
vp[5] = new Revista("Publicatia6", 4, "Revista3", 1000);

cout << "\n\nVectorul cu 3 publicatii si 3 reviste:" << endl;
for (int i = 0; i < n; i++)
{
    vp[i]->afisare();
}
```

7p+1.5p=8.5p

TABLOUL DE POINTERI LA CLASA PUBLICAȚIE [POLIMORFISMUL]

```
int n = 6;
Publicatie** vp = new Publicatie*[n];
vp[0] = new Publicatie("Publicatia1", 3);
vp[1] = new Publicatie("Publicatia2", 30);
vp[2] = new Publicatie("Publicatia3", 300);
vp[3] = new Revista("Publicatia4", 1, "Revist
vp[4] = new Revista("Publicatia5", 2, "Revist
vp[5] = new Revista("Publicatia6", 4, "Revist

cout << "\n\nVectorul cu 3 publicatii si 3 re
for (int i = 0; i < n; i++)
{
    vp[i]->afisare();
```

Publicatie

```
#pragma once

class Publicatie
{
    string Titlu;
    unsigned int NrPagini;
public:
    Publicatie();
    //Publicatie(Publicatie &p);
    Publicatie(string t, unsigned int np);
    ~Publicatie();
    virtual void afisare();
    string getTitlu();
```

- Testați metoda de afișare utilizând numai obiectele din acest tablou:
  - se va avea în vedere polimorfismul (aplicația trebuie să asigure faptul că pentru obiectele de tip Publicatie se va apela metoda din clasa Publicatie, iar pentru obiectele de tip Revista se va apela metoda din clasa Revista)

PENTRU NOTA >=9

CALCULAREA ȘI AFIȘAREA TIRAJULUI TOTAL

- Calculați și afișați tirajul total al celor 3 reviste.

```
unsigned int tirajTotal(Publicatie** vp, int n)
{
    Revista* r;
    unsigned int totalTiraj = 0;
    for (int i = 0; i < n; i++)
    {
        r = dynamic_cast<Revista*>(vp[i]);
        if (r)
        {
            r->afisare();
            totalTiraj += r->getTiraj();
        }
    }
    return totalTiraj;
}
```

```
unsigned int Revista::getTiraj()
{
    return Tiraj;
}
```

```
cout << "\n\nTirajul total: " << tirajTotal(vp,n) << endl;
```

8.5p+0.5p=9.0p

SORTAREA TABLOULUI DUPĂ NUMĂRUL DE PAGINI SORTAREA CLASICĂ

- Sortați descendent după numărul de pagini obiectele din tablou (bonus 0.5p pentru utilizarea funcției sort)

```
void sortareClasica(Publicatie** &vp, int n)
{
    Publicatie* aux;
    for (int i = 0; i < n - 1; i++)
    for (int j = i + 1; j < n; j++)
    if (vp[i]->getNrPagini() < vp[j]->getNrPagini())
    {
        aux = vp[i];
        vp[i] = vp[j];
        vp[j] = aux;
    }

}
```

```
unsigned int Publicatie::getNrPagini()
{
    return NrPagini;
}
```

```
//sortarea fara bonus
sortareClasica(vp, n);
cout << "\n\n\t Dupa sortarea clasica: " << endl;
for (int i = 0; i < n; i++)
{
    vp[i]->afisare();
}
```

9p+1p + 0.5p=10.5p

SORTAREA TABLOULUI DUPĂ NUMĂRUL DE PAGINI SORTAREA CU METODA SORT [ALGORITHM]

- Sortați descendent după numărul de pagini obiectele din tablou (bonus 0.5p pentru utilizarea funcției sort)

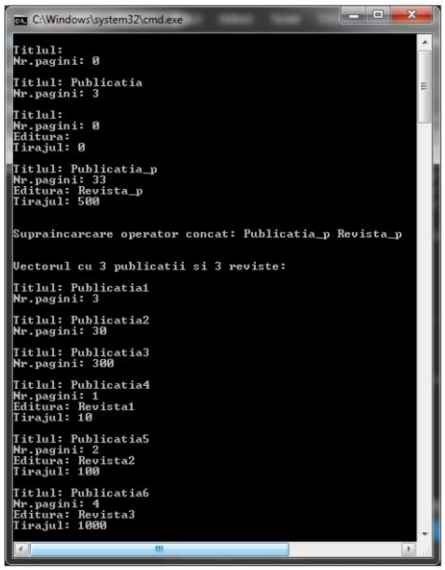
```
void sortareSort(vector<Publicatie> &svp)
{
    sort(svp.begin(), svp.end());
    reverse(svp.begin(), svp.end());
}
```

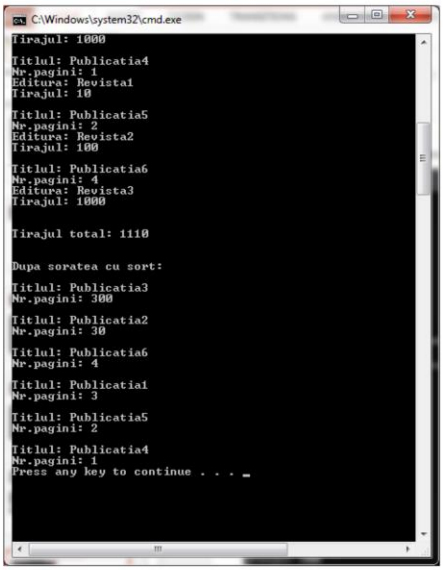
```
bool Publicatie::operator<(const Publicatie &b)
{
    return (NrPagini < b.NrPagini);
}
```

```
#include <vector>
#include <algorithm>
```

```
//sortarea cu bonus
vector<Publicatie> svp;
for (int i = 0; i < n; i++)
{
    svp.push_back(*vp[i]);
}
sortareSort(svp);
cout<<"\n\nDupa soratea cu sort: "<<endl;
for (int i = 0; i < n; i++)
{
    svp[i].afisare();
}
```

9p+1.5p + 0.5p=11p


C:\Windows\system32\cmd.exe
Titlul:
Nr.pagini: 0
Titlul: Publicatia
Nr.pagini: 3
Titlul:
Nr.pagini: 0
Editura:
Tirajul: 0
Titlul: Publicatia_p
Nr.pagini: 33
Editura: Revista_p
Tirajul: 500
Supraincarcare operator concat: Publicatia_p Revista_p
Vectorul cu 3 publicatii si 3 reviste:
Titlul: Publicatia1
Nr.pagini: 3
Titlul: Publicatia2
Nr.pagini: 30
Titlul: Publicatia3
Nr.pagini: 300
Titlul: Publicatia4
Nr.pagini: 1
Editura: Revista1
Tirajul: 10
Titlul: Publicatia5
Nr.pagini: 2
Editura: Revista2
Tirajul: 100
Titlul: Publicatia6
Nr.pagini: 4
Editura: Revista3
Tirajul: 1000


C:\Windows\system32\cmd.exe
Tirajul: 1000
Titlul: Publicatia4
Nr.pagini: 1
Editura: Revista1
Tirajul: 10
Titlul: Publicatia5
Nr.pagini: 2
Editura: Revista2
Tirajul: 100
Titlul: Publicatia6
Nr.pagini: 4
Editura: Revista3
Tirajul: 1000
Tirajul total: 1110
Dupa soratea cu sort:
Titlul: Publicatia3
Nr.pagini: 300
Titlul: Publicatia2
Nr.pagini: 30
Titlul: Publicatia6
Nr.pagini: 4
Titlul: Publicatia1
Nr.pagini: 3
Titlul: Publicatia5
Nr.pagini: 2
Titlul: Publicatia4
Nr.pagini: 1
Press any key to continue . . .