**Operatorul de rezolutie si alocare dinamica:**

Un operator de rezolutie ( :: ) permite accesul la un identificator cu domeniu fisier, dintr-un bloc în care acesta nu este vizibil, datorita unei alte declaratii.

#include <iostream>

using namespace std;

void afiseaza(int n) {

cout << "Se afiseaza n = " << n << endl;

}

int n = 1; //variabila globala

int main() {

int\* ip1, \* ip2, \* ip3;

ip1 = new int; //valoare intreaga neinitializata, alocare dinamica cu operatorul

//new

ip2 = new int(2); //valoare intreaga initializata cu 2

ip3 = new int[100]; //tablou de 100 de intregi

int n = 2; //variabila locala

afiseaza(n); //afiseaza 2, valoarea variabilei locale n

afiseaza(::n); //afiseaza 1, valoarea variabilei gloale n cu ajutorul operatorului

//de rezolutie

cout << "S-a executat return 0";

return 0;

}

**Operatori de selectie . si ->**

 Principala utilizare a lui this este la scrierea functiilor care manipuleza direct pointeri

#include <iostream>

class Operatori {

private:

public:

void push(int);

};

void push(int x) {

cout << "Mesaj din metoda push ce are valoarea: "<< x <<endl;

}

using namespace std;

int main() {

Operatori s1, \* ps;

ps = new Operatori;

s1.push(5); // . se numeste operator de selectie (se foloseste pentru obiecte

//alocate static)

ps->push(8); // -> se numeste operator de selectie (se foloseste pentru obiecte

//alocate dinamic)

cout << "S-a executat return 0";

return 0;

}

**Functiile *prietene* (friend)** pot folosi membrii privati ai unei clase, desi ele însele nu sunt membri. Functiile prietene au fost introduse pentru cazurile în care o functie coopereaza strâns cu o clasa, dar nu face parte din acea clasa. O functie prietena este declarata asemanator cu o functie membru, având prototipul în interiorul clasei, precedat de cuvântul cheie friend. Exista si posibilitatea ca una sau mai multe functii membre ale unei clase sa fie functii prietene ale altei clase. Se poate declara si o clasa Y prietena a unei clase X, în acest caz toate functiile clasei Y sunt prietene ale clasei X si au acces nelimitat la membrii privati ai clasei X.

#include <iostream>

using namespace std;

class Operatori {

private:

protected:

int varf;

public:

void push(int);

friend class Operator2;

};

class Operator2 {

public:

void metoda();

};

void Operatori::push(int x) {

cout << "Mesaj din metoda push ce are valoarea: " << x << endl;

}

void Operator2::metoda() {

Operatori s;

s.varf = 5; //devine o operatie permisa pentru ca Operator e clasa prietena

cout << "S-a apelat functia numita metoda care e prietena clasei Operatori si are acces la atributul varf din protected: " << s.varf << endl;

}

int main() {

Operatori s1, \* ps;

Operator2 s2;

ps = new Operatori;

s1.push(5); // . se numeste operator de selectie (se foloseste pentru obiecte alocate static)

ps->push(8); // -> se numeste operator de selectie (se foloseste pentru obiecte alocate dinamic)

s2.metoda();

cout << "S-a executat return 0";

return 0;

}

**Constuctori si destructori**

Pentru crearea, initializarea, copierea si respectiv distrugerea obiectelor, în C++ se folosesc functii speciale, numite constructori si destructori. Constructorul se apeleaza automat la crearea fiecarui obiect al clasei, static, automatic sau dinamic(cu operatorul new), inclusiv pentru obiecte temporare.

Destructorul este apelat automat la eliminarea unui obiect, la încheierea timpului sau de viata, sau poate fi solicitat prin program, cu operatorul delete.

Constructorii si destructorii se declara si se definesc similar cu celelalte functii membre, dar se disting de acestea printr-o serie de caracteristici specifice:

* numele functiilor constructor sau destructor coincide cu numele clasei careia îi apartin; destructorii au numele precedat de caracterul (~).
* în declaratie si definitie nu se specifica nici un tip de rezultat, nici macar void
* constructorii pot avea parametrii, inclusiv parametrii impliciti, si pot fi supradefiniti; destructorii nu au aceste proprietati
* nu se pot utiliza pointeri catre constructori sau destructori
* daca o clasa nu dispune de constructori si destructori definiti, compilatorul va genera automat un constructor implicit, respectiv un destructor, functii publice

**Intrebari, cautari, nelamuriri:**

Supraincarcarea operatorilor ???