**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**RAPORT**

Lucrare de laborator nr. 2

la cursul ***„Programarea orientată pe obiecte”***

***Tema: Supraîncărcarea operatorilor***

**Varianta 21**

**A efectuat :**   **St. gr. CR-221FR Serba Cristina**

**A verificat: asist. univ. Mantaluța Marius**

**Chișinău 2024**

# Sarcina lucrării:

Să se creeze clasa 2-D de coordonate de tip double în plan. Să se definească operatorii "+" şi "-

" ca funcţii prietene, iar operatorii de atribuire şi de comparare – ca metode ale clasei. De

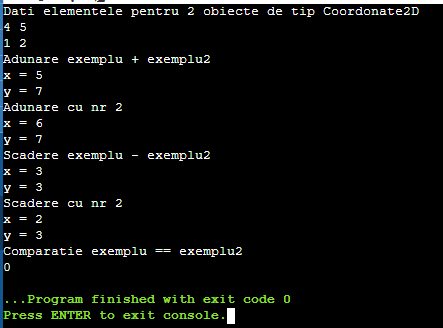
prevăzut posibilitatea efectuării operaţiilor atît între coordonate, cît şi între coordonate şi

numere obişnuite. Să se definească operatorii "<<" şi ">>"

# Mersul programului:

|  |
| --- |
| #include <iostream>  using namespace std;  class Coordonate2D  {  public:  double x;  double y;    Coordonate2D()  {  x = 0;  y = 0;  }    Coordonate2D(double xParam, double yParam)  {  x = xParam;  y = yParam;  }    friend Coordonate2D operator +(const Coordonate2D &first, const Coordonate2D &second);  friend Coordonate2D operator +(const Coordonate2D &example, const double factor);  friend Coordonate2D operator -(const Coordonate2D &first, const Coordonate2D &second);  friend Coordonate2D operator -(const Coordonate2D &example, const double factor);    void operator =(Coordonate2D &example)  {  x = example.x;  y = example.y;  }    bool operator ==(Coordonate2D &second)  {  if ((x == second.x) && (y == second.y))  {  return true;  }  else  {  return false;  }  }    friend ostream& operator <<(ostream &out, Coordonate2D &example);  friend istream& operator >>(istream &in, Coordonate2D &example);  };  Coordonate2D operator +(const Coordonate2D &first, const Coordonate2D &second)  {  Coordonate2D result;  result.x = first.x + second.x;  result.y = first.y + second.y;  return result;  }  Coordonate2D operator -(const Coordonate2D &first, const Coordonate2D &second)  {  Coordonate2D result;  result.x = first.x - second.x;  result.y = first.y - second.y;  return result;  }  Coordonate2D operator +(const Coordonate2D &example, const double factor)  {  Coordonate2D result;  result.x = example.x + factor;  result.y = example.y + factor;  return result;  }  Coordonate2D operator -(const Coordonate2D &example, const double factor)  {  Coordonate2D result;  result.x = example.x - factor;  result.y = example.y - factor;  return result;  }  ostream& operator <<(ostream &out, Coordonate2D &example)  {  out << "x = " << example.x << endl;  out << "y = " << example.y << endl;  return out;  }  istream& operator >>(istream &in, Coordonate2D &example)  {  in >> example.x;  in >> example.y;  return in;  }  int main()  {  Coordonate2D exemplu;  Coordonate2D exemplu2;    cout << "Dati elementele pentru 2 obiecte de tip Coordonate2D\n";  cin >> exemplu;  cin >> exemplu2;    Coordonate2D result = exemplu + exemplu2;  cout << "Adunare exemplu + exemplu2\n" << result;    Coordonate2D result2 = exemplu + 2.0;  cout << "Adunare cu nr 2\n" << result2;    Coordonate2D result3 = exemplu - exemplu2;  cout << "Scadere exemplu - exemplu2\n" << result3;    Coordonate2D result4 = exemplu - 2.0;  cout << "Scadere cu nr 2\n" << result4;    bool resultComparation = exemplu == exemplu2;  cout << "Comparatie exemplu == exemplu2\n" << resultComparation;    return 0;  } |

# Rezultatele obținute:



# Concluzii:

În concluzie, supraincarcarea operatorilor în C++ este o caracteristică puternică care permite programatorilor să extindă funcționalitatea operatorilor încorporați către tipuri definite de utilizator. Prin redefinirea comportamentului operatorilor, dezvoltatorii pot face codul lor mai intuitiv, expresiv și concis. Cu toate acestea, este crucial să folosiți supraincarcarea operatorilor cu discernământ și cu atenție, deoarece utilizarea greșită poate duce la confuzie și comportament neintenționat. Atunci când este aplicată cu grijă, supraincarcarea operatorilor poate îmbunătăți semnificativ lizibilitatea și ușurința de întreținere a codului C++, deblocând întregul potențial al paradigmelor de programare orientate pe obiecte.