# Proiect: Aplicații C++ pentru Proba Practică

Proiect: Aplicații C++ pentru Proba Practică

Acest proiect conține mai multe aplicații C++ separate, fiecare realizând una dintre cerințele specificate în proba practică. Fiecare aplicație este explicată pas cu pas pentru a facilita înțelegerea funcționalității și implementării.

1. Declararea și utilizarea variabilelor de orice tip

---------------------------------------------------

#include <iostream>  
using namespace std;  
  
int main() {  
 // Declararea variabilelor  
 int varInt = 0; // Variabilă întreagă inițializată cu 0  
 float varFloat = 0.0; // Variabilă float inițializată cu 0.0  
 char varChar = 'A'; // Variabilă caracter inițializată cu 'A'  
  
 // Afișarea valorilor variabilelor  
 cout << "Valoarea lui varInt: " << varInt << endl;  
 cout << "Valoarea lui varFloat: " << varFloat << endl;  
 cout << "Valoarea lui varChar: " << varChar << endl;  
  
 return 0;  
}

Explicații:

- Se declară trei variabile de tip int, float și char.

- Fiecare variabilă este inițializată cu o valoare implicită și afișată în consolă.

2. Preluarea și afișarea datelor de la tastatură

------------------------------------------------

#include <iostream>  
using namespace std;  
  
int main() {  
 int varInt;  
 cout << "Introduceți un număr întreg: ";  
 cin >> varInt; // Citește un număr întreg de la tastatură  
 cout << "Ați introdus: " << varInt << endl; // Afișează numărul introdus  
  
 return 0;  
}

Explicații:

- Se solicită utilizatorului să introducă un număr întreg.

- Numărul introdus este citit folosind `cin` și apoi afișat folosind `cout`.

3. Utilizarea unei instrucțiuni if-else

---------------------------------------

#include <iostream>  
using namespace std;  
  
int main() {  
 int varInt;  
 cout << "Introduceți un număr întreg: ";  
 cin >> varInt; // Citește un număr întreg de la tastatură  
  
 // Verificarea valorii lui varInt  
 if (varInt > 10) {  
 cout << "Numărul este mai mare decât 10." << endl;  
 } else {  
 cout << "Numărul este mai mic sau egal cu 10." << endl;  
 }  
  
 return 0;  
}

Explicații:

- Se verifică dacă numărul introdus de utilizator este mai mare sau mai mic decât 10 folosind o instrucțiune if-else.

4. Utilizarea unei bucle for

----------------------------

#include <iostream>  
using namespace std;  
  
int main() {  
 // Buclă for pentru a afișa mesajul de mai multe ori  
 for (int i = 0; i < 5; i++) {  
 cout << "Aceasta este iterația " << i+1 << endl;  
 }  
  
 return 0;  
}

Explicații:

- O buclă for este utilizată pentru a afișa un mesaj de 5 ori, incrementând un contor de la 0 la 4.

5. Definirea și apelarea funcțiilor

-----------------------------------

#include <iostream>  
using namespace std;  
  
// Funcție fără parametri care afișează un mesaj  
void mesaj() {  
 cout << "Aceasta este o funcție fără parametri." << endl;  
}  
  
// Funcție care calculează suma a două numere întregi și returnează rezultatul  
int suma(int a, int b) {  
 return a + b;  
}  
  
int main() {  
 mesaj(); // Apelarea funcției fără parametri  
 cout << "Suma celor două numere este: " << suma(5, 10) << endl; // Apelarea funcției suma  
  
 return 0;  
}

Explicații:

- Se definesc două funcții: `mesaj`, care afișează un mesaj, și `suma`, care calculează suma a două numere.

- Ambele funcții sunt apelate în funcția `main`.

6. Definirea și utilizarea unui tablou

--------------------------------------

#include <iostream>  
using namespace std;  
  
int main() {  
 // Definirea unui tablou și afișarea valorilor din acesta  
 int tablou[5] = {1, 2, 3, 4, 5}; // Tablou de numere întregi  
 for (int i = 0; i < 5; i++) {  
 cout << "Valoarea la indexul " << i << " este: " << tablou[i] << endl;  
 }  
  
 return 0;  
}

Explicații:

- Se declară un tablou de întregi și se afișează fiecare element folosind o buclă for.

7. Definirea și utilizarea unei structuri

-----------------------------------------

#include <iostream>  
#include <cstring>  
using namespace std;  
  
// Structură pentru reprezentarea unui student  
struct Student {  
 char nume[50]; // Numele studentului  
 int nota; // Nota studentului  
};  
  
int main() {  
 // Definirea și utilizarea unei structuri  
 Student student;  
 strcpy(student.nume, "Ion Popescu"); // Setarea numelui studentului  
 student.nota = 10; // Setarea notei studentului  
 cout << "Student: " << student.nume << ", Nota: " << student.nota << endl; // Afișarea detaliilor studentului  
  
 return 0;  
}

Explicații:

- Se definește o structură pentru a reprezenta un student și se afișează detaliile acestuia.

8. Utilizarea unui pointer

--------------------------

#include <iostream>  
using namespace std;  
  
int main() {  
 int varInt = 10;  
 int \*ptr = &varInt; // Pointer care memorează adresa lui varInt  
 cout << "Adresa lui varInt: " << ptr << endl; // Afișarea adresei lui varInt  
 cout << "Valoarea la care pointează ptr: " << \*ptr << endl; // Afișarea valorii la care pointează ptr  
  
 return 0;  
}

Explicații:

- Se declară un pointer care memorează adresa unei variabile și se afișează adresa și valoarea la care pointează pointerul.

9. Modificarea valorii prin pointer

-----------------------------------

#include <iostream>  
using namespace std;  
  
int main() {  
 int varInt = 10;  
 int \*ptr = &varInt; // Pointer care memorează adresa lui varInt  
 \*ptr = 20; // Modificarea valorii variabilei prin pointer  
 cout << "Noua valoare a variabilei este: " << varInt << endl; // Afișarea noii valori a lui varInt  
  
 return 0;  
}

Explicații:

- Se modifică valoarea unei variabile folosind un pointer și se afișează noua valoare.

10. Alocarea și eliberarea memoriei

-----------------------------------

#include <iostream>  
#include <cstdlib>  
using namespace std;  
  
int main() {  
 // Alocarea dinamică a memoriei  
 int \*dinamic = (int \*)malloc(sizeof(int) \* 5); // Alocare dinamică pentru un tablou de 5 întregi  
 if (dinamic == NULL) {  
 cout << "Alocarea memoriei a eșuat." << endl;  
 return 1; // Întoarce 1 în caz de eroare  
 }  
  
 // Eliberarea memoriei alocate dinamic  
 free(dinamic); // Eliberarea memoriei alocate  
  
 return 0;  
}

Explicații:

- Se alocă și eliberează dinamic memorie pentru un tablou de întregi folosind `malloc` și `free`.