ISSN: 2181-3906 2025

International scientific journal
«MODERN SCIENCE AND RESEARCH»
VOLUME 4 / ISSUE 11 / UIF:8.2 / MODERNSCIENCE.UZ

C++ DA OBJECT-ORIENTED PROGRAMMING (OOP) TAMOYILLARINING AMALIY QO‘LLANILISHI

Ramazonova O’g’ilshod Rustam qizi
Termiz davlat univeristeti 3-kurs talabasi.

https://doi.org/10.5281/zenodo.17761141

Annotatsiya. *Ushbu maqolada C++ dasturlash tilida obyektga yo‘naltirilgan dasturlash (Object-Oriented Programming – OOP) tamoyillari va ularning amaliy qo‘llanilishi yoritilgan. OOP dasturlash texnologiyalarining muhim yo‘nalishlaridan biri bo‘lib, dastur tuzilmasini modul shaklida tashkil etish, kodni qayta ishlatish va murakkab dasturiy tizimlarni soddalashtirish imkonini beradi. Maqolada OOP ning asosiy to‘rt tamoyili – inkapsulyatsiya (encapsulation), meros olish (inheritance), polimorfizm (polymorphism) va abstraksiya (abstraction) nazariy va amaliy misollar bilan yoritilgan. Shuningdek, C++ tilida klasslar (classes) va obyektlar (objects) orqali real hayotdagi jarayonlarni modellashtirish usullari ham ko‘rib chiqiladi.*

Kalit so’zlar: *C++ dasturlash tili, obyektga yo‘naltirilgan dasturlash, OOP, klass, obyekt, inkapsulyatsiya, meros olish, polimorfizm, abstraksiya, konstruktor, destruktor, metod, kapsulalash, qayta foydalanish, modular dasturlash, real hayot modellashtirish, dastur tuzilmasi.*

Abstarct. *This article discusses the principles of Object-Oriented Programming (OOP) in the C++ programming language and their practical application. OOP is one of the important areas of programming technologies, which allows organizing the program structure in a modular form, reusing code, and simplifying complex software systems. The article discusses the four main principles of OOP - encapsulation, inheritance, polymorphism, and abstraction - with theoretical and practical examples. It also discusses methods for modeling real-life processes using classes and objects in the C++ language.*

Keywords: *C++ programming language, object-oriented programming, OOP, class, object, encapsulation, inheritance, polymorphism, abstraction, constructor, destructor, method, encapsulation, reuse, modular programming, real-life modeling, program structure.*

C++ dasturlash tilida obyektga yo‘naltirilgan dasturlash (OOP) dasturiy tizimlarni modul tarzida yaratish, kodni qayta ishlatish va murakkab loyihalarni boshqarishni osonlashtiruvchi yondashuv hisoblanadi. OOP tamoyillari orqali dastur real hayotdagi obyektlar kabi tuziladi, ularning xususiyatlari va vazifalari aniqlanadi. Masalan, avtomobil obyektining rang, model va tezlik xususiyatlari bo‘lishi mumkin va u yurish, to‘xtash va signal berish kabi funksiyalarni bajaradi.

C++ tilida obyektlar klasslar orqali yaratiladi va klass ma’lumotlar va metodlarni bir joyda birlashtiruvchi shablondir. Shu maqsadda Car klassini yaratib, unga brand va speed xususiyatlarini berish va drive() funksiyasini qo‘shish mumkin:

```
#include <iostream>
using namespace std;
class Car {
public:
```

```
string brand;
int speed;
void drive() {
cout << brand << " mashinasi " << speed << " km/h tezlikda harakat qilmoqda" << endl;
}
};

int main() {
Car car1;
car1.brand = "BMW";
car1.speed = 120;
car1.drive();
return 0;
```

}orqali obyekt yaratib mashina harakatini simulyatsiya qilish mumkin.

OOP tamoyillaridan biri inkapsulyatsiya bo‘lib, ma’lumotlar va ularni ishlovchi funksiyalarni bitta birlik ichida saqlashni anglatadi. Bu tashqi koddan ma’lumotlarni bevosita o‘zgartirishga yo‘l qo‘ymaydi va dastur xavfsizligini oshiradi. Masalan, BankAccount klassida balans qiymatini faqat deposit va withdraw funksiyalari orqali o‘zgartirish mumkin:

```
#include <iostream>
using namespace std;
class BankAccount {
private:
double balance;
public:
void deposit(double amount) { balance += amount; }
void withdraw(double amount) { if(amount <= balance) balance -= amount; }
double getBalance() { return balance; }
};
int main() {
BankAccount account1;
account1.deposit(500);
account1.withdraw(200);
cout << "Balans: " << account1.getBalance() << endl;
return 0;
}
Shu bilan
BankAccount account1;
account1.deposit(500);
account1.withdraw(200);
cout << "Balans: " << account1.getBalance();
```

orqali foydalanuvchi balansni xavfsiz boshqarishi mumkin.

Meros olish orqali umumiy xususiyatlar bir klassda saqlanadi va undan boshqa klasslarda foydalanish mumkin. Masalan, Person klassidan Teacher klassi meros oladi:

```
#include <iostream>
using namespace std;
class Person {
public:
string name;
int age;
void displayInfo() {
cout << "Ism: " << name << ", Yosh: " << age << endl;
}
};
class Teacher : public Person {
public:
string subject;
void displayTeacherInfo() {
displayInfo();
cout << "Fani: " << subject << endl;
}
};
```

Shu tarzda

```
int main() {
Teacher teacher1;
teacher1.name = "Sara";
teacher1.age = 35;
teacher1.subject = "Matematika";
teacher1.displayTeacherInfo();
return 0;
```

}orqali o‘qituvchi obyektini yaratish va uning ma’lumotlarini ekranga chiqarish mumkin.Polimorfizm bir xil nomdagi funksiyaning turli klasslarda turlicha ishlashini ta’minlaydi.

Masalan, Animal klassida virtual makeSound() funksiyasini belgilash va Dog hamda Cat klasslarida o‘zicha ishlatish mumkin:

```
#include <iostream>
using namespace std;
class Animal {
public:
virtual void makeSound() { cout << "Ovoz" << endl; }
};
class Dog : public Animal {
public:
void makeSound() override { cout << "Vov" << endl; }
```

```
};
class Cat : public Animal {
public:
void makeSound() override { cout << "Miyav" << endl; }
};
int main() {
Animal* a1 = new Dog();
a1->makeSound();
Animal* a2 = new Cat();
a2->makeSound();
delete a1;
delete a2;
return 0;
```

}kodlari orqali turli hayvonlar uchun ovoz chiqarish funksiyasini chaqirish mumkin.

Abstraksiya foydalanuvchiga faqat kerakli funksiyalarni ko‘rsatadi va ichki murakkab jarayonlarni yashiradi. Masalan, haydovchi mashinani boshqarayotganda dvigatel qanday ishlashini bilmaydi, faqat gaz va tormoz pedallarini ishlatadi. Shu tamoyil dasturiy tizimlarda foydalanuvchiga qulay interfeys yaratishda yordam beradi.

Konstruktorlar obyekt yaratilganda avtomatik chaqiriladi va uning xususiyatlarini boshlang‘ich qiymatlar bilan to‘ldiradi, destruktorlar esa obyekt yo‘q qilinayotganda ishlaydi.

Masalan, Book klassida konstruktor orqali kitobning nomi va muallifi belgilanadi:

```
#include <iostream>
using namespace std;
class Book {
string title;
string author;
public:
Book(string t, string a) {
title = t;
author = a;
cout << "Kitob yaratildi: " << title << endl;
}
void displayBookInfo() {
cout << "Nom: " << title << ", Muallif: " << author << endl;
}
~Book() {
cout << "Kitob obyekti yo'q qilindi: " << title << endl;
}
};
int main() {
Book book1("C++ Dasturlash", "Ali Rustamov");
book1.displayBookInfo();
```

```
return 0;
}
```

orqali foydalanuvchi obyektni yaratadi va uni ishlatadi.

OOP tamoyillari amaliy hayotda keng qo‘llaniladi. Masalan, kutubxona tizimida har bir kitob obyekt sifatida saqlanadi, foydalanuvchi kitob qo‘shish, o‘chirish yoki qarzga olish imkoniyatiga ega bo‘ladi. Bank tizimida har bir hisob raqami obyekt bo‘lib, foydalanuvchi depozit qilish, pul yechish yoki balansni ko‘rish funksiyalarini bajaradi. Hayvonlar bilan ishlaydigan dasturda esa umumiy Animal klassi asosida Dog va Cat klasslari yaratiladi va ular o‘z xatti-harakatlarini bajaradi. Shu tarzda dastur modul va tushunarli bo‘ladi, kodni qayta ishlatish imkoniyati oshadi va murakkab tizimlarni boshqarish soddalashtiriladi.

Natijada, C++ da obyektga yo‘naltirilgan dasturlash tamoyillarini tushunish va amaliyotda qo‘llash dasturchiga professional darajada kod yozish imkonini beradi, dasturiy tizimlarni modul va xavfsiz tarzda yaratishga yordam beradi, murakkab loyihalarni boshqarish osonlashadi, hamda real hayotiy jarayonlar dastur ichida to‘liq modellashtiriladi.

Xulosa

C++ dasturlash tilida obyektga yo‘naltirilgan dasturlash (OOP) tamoyillari dasturchilarga murakkab dasturiy tizimlarni mantiqiy bloklarga ajratish, kodni qayta ishlatish va xatoliklarni kamaytirish imkonini beradi. OOP asosiy tamoyillari — inkapsulyatsiya, meros olish, polimorfizm va abstraksiya — dasturiy loyihalarni modul va xavfsiz shaklda yaratishga yordam beradi. Inkapsulyatsiya yordamida ma’lumotlar va ularni qayta ishlovchi funksiyalar bir joyda saqlanadi, bu esa dastur xavfsizligini oshiradi va foydalanuvchining noto‘g‘ri xatti-harakatlaridan himoya qiladi. Masalan, BankAccount klassida balans qiymati faqat deposit va withdraw funksiyalari orqali o‘zgartiriladi, foydalanuvchi to‘g‘ridan-to‘g‘ri balansni o‘zgartira olmaydi. C++ da obyektga yo‘naltirilgan dasturlash dasturchiga kodni modul, xavfsiz va samarali yozish imkonini beradi, real hayotiy jarayonlarni dasturiy tizimga modellashtirishni osonlashtiradi va murakkab tizimlarni boshqarishni sezilarli darajada soddalashtiradi. Shu bilan birga, amaliy misollar orqali foydalanuvchi obyektlar yaratishni, ularni boshqarishni va metodlar yordamida ma’lumotlarni ishlatishni aniq tushunadi. OOP tamoyillarini o‘rganish dasturchiga professional darajada loyihalarni yaratish imkoniyatini beradi va dasturiy yechimlarni samarali va barqaror qiladi.

Foydalanilgan adabiyotlar

1. Q. S. Raxmanov, D. S. Tuxtanazarov. *Dasturlash I. C++ dasturlash tili*. Toshkent – “Zamon Poligraf” nashriyoti, 2023-yil. tiu-edu.uz
2. “C++ dasturlash tili” – O‘zbek tilida qo‘llanma. Renessans-Edu.uz, Toshkent, 2023-yil. renessans-edu.uz
3. “C++ da OOP (Object Oriented Programming)”; O‘zbek tilida onlayn darslik. UzbekDevs.uz.