

PRZEDMIOT: Podstawy programowania

KLASA: 2A gr. 1

Lekcja

Temat: Klasy w C++

Klasa w C++ to podstawowy element programowania obiektowego (OOP), który pozwala na definiowanie własnych typów danych. Jest to szablon lub blueprint, który łączy w sobie dane (pola lub zmienne członkowskie) oraz funkcje (metody), które operują na tych danych. Klasy umożliwiają enkapsulację (ukrywanie szczegółów implementacji), dziedziczenie (ponowne wykorzystanie kodu) i polimorfizm (różne zachowania w zależności od kontekstu).

Przykład 1:

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

class Samochod {
private:
    string marka;
    int rokProdukcji;

public:
    Samochod(std::string m, int r) {
        marka = m;
        rokProdukcji = r;
    }
    void wyswietlInfo() {
        cout << "Marka: " << marka << ", Rok produkcji: " << rokProdukcji << endl;
    }
};

int main() {
    Samochod mojSamochod("Toyota", 2020);
    mojSamochod.wyswietlInfo();
    return 0;
}
```

Kluczowe cechy klas w C++

- **Konstruktor i destruktor:** Konstruktor (np. `Samochod()`) inicjalizuje obiekt, destruktor (`~Samochod()`) czyści zasoby po zniszczeniu obiektu.
- **Dziedziczenie:** Możesz tworzyć klasy pochodne, np. `class ElektrycznySamochod : public Samochod { ... };`
- **Polimorfizm:** Metody wirtualne (`virtual`) pozwalają na nadpisywanie zachowań w klasach dziedziczących.
- **Statyczne elementy:** Pola lub metody statyczne (`static`) należą do klasy, nie do instancji (np. licznik obiektów).

Klasy są podobne do struktur (`struct`), ale domyślnie w `struct` elementy są publiczne, a w `class` prywatne. W praktyce klasy są używane do modelowania rzeczywistych obiektów, co ułatwia pisanie modularnego i utrzymywalnego kodu

Konstruktor

Konstruktor to specjalna metoda klasy, która jest wywoływana automatycznie w momencie tworzenia obiektu (instancji klasy). Jego głównym zadaniem jest inicjalizacja pól klasy, alokacja zasobów (np. pamięci) lub ustawienie początkowych wartości. Konstruktor ma taką samą nazwę jak klasa i nie zwraca żadnej wartości (nawet `void` nie jest potrzebne).

- **Rodzaje konstruktorów:**
 - **Domyślny:** Bez parametrów, tworzony automatycznie przez kompilator, jeśli nie zdefiniujesz własnego.
 - **Parametryzowany:** Z parametrami, jak w przykładzie poniżej.
 - **Kopiujący:** Kopiuje dane z innego obiektu.
 - **Przenoszący** (w C++11+): Przenosi zasoby z innego obiektu.

Jeśli nie zdefiniujesz konstruktora, kompilator stworzy domyślny.

Destruktor

Destruktor to specjalna metoda klasy, która jest wywoływana automatycznie w momencie niszczenia obiektu (np. gdy obiekt wychodzi poza zakres widoczności lub jest usuwany ręcznie za pomocą `delete`). Służy do zwalniania zasobów, takich jak pamięć, pliki czy połączenia sieciowe, aby uniknąć wycieków pamięci. Destruktor ma nazwę klasy poprzedzoną tyldą (`~`) i nie przyjmuje parametrów ani nie zwraca wartości.

- Klasa może mieć tylko jeden destruktor.
- Jeśli nie zdefiniujesz destruktora, kompilator stworzy domyślny, który nic nie robi (chyba że klasa ma pola wymagające czyszczenia).
- Destruktry są szczególnie ważne w klasach zarządzających zasobami (np. wskaźnikami).

Przykład kodu

Przykład 2:

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

class Samochod {
private:
    string marka;
    int rokProdukcji;
    int* numerVIN; // Przykładowe dynamiczne alokowanie pamięci

public:
    // Konstruktor parametryzowany
    Samochod(string m, int r) {
        marka = m;
        rokProdukcji = r;
        numerVIN = new int; // Alokacja pamięci
        *numerVIN = 123456;
        std::cout << "Konstruktor wywołany: Obiekt stworzony." << endl;
    }

    // Destraktor
    ~Samochod() {
        delete numerVIN; // Zwolnienie pamięci, aby uniknąć wycieku
        cout << "Destraktor wywołany: Obiekt zniszczony." << endl;
    }

    void wyswietlInfo() {
        cout << "Marka: " << marka << ", Rok produkcji: " << rokProdukcji
            << ", Numer VIN: " << *numerVIN << endl;
    }
};

int main() {
    {
        Samochod mojSamochod("Toyota", 2020);
        mojSamochod.wyswietlInfo();
    } // Koniec bloku - obiekt niszczone, destraktor wywołany automatycznie

    return 0;
}
```

Wyjaśnienie przykładu:

- **Konstruktor:** Samochod(std::string m, int r) inicjalizuje pola marka i rokProdukcji, alokuje pamięć dla numerVIN i wyświetla komunikat.
- **Destraktor:** ~Samochod() zwalnia pamięć za pomocą delete i wyświetla komunikat. Wywoływany automatycznie na końcu bloku {} w main().

Bez destruktora pamięć po numerVIN nie zostałaby zwolniona, co mogłoby prowadzić do problemów w większych programach.

Przykład 3:

```
#include <iostream>
#include <string>

class Osoba {
private:
    std::string imie;
    std::string nazwisko;
    int wiek;

public:
    Osoba(std::string i, std::string n, int w) {
        imie = i;
        nazwisko = n;
        wiek = w;
    }

    void wyswietlInfo() {
        std::cout << "Imię: " << imie << ", Nazwisko: " << nazwisko << ", Wiek: " << wiek
<< std::endl;
    }

    // Metoda sprawdzająca pełnoletniość (przykład logiki)
    bool jestPelnoletnia() {
        return (wiek >= 18);
    }
};

int main() {
    Osoba mojaOsoba("Jan", "Kowalski", 25);
    mojaOsoba.wyswietlInfo();

    if (mojaOsoba.jestPelnoletnia()) {
        std::cout << "Osoba jest pełnoletnia." << std::endl;
    } else {
        std::cout << "Osoba nie jest pełnoletnia." << std::endl;
    }

    return 0;
}
```

Przykład 4:

```
#include <iostream>
#include <string>

class Dom {
private:
    std::string adres;
    int liczbaPokoi;
    double powierzchnia; // w metrach kwadratowych

public:
    // Konstruktor parametryzowany
    Dom(std::string a, int p, double pow) {
        adres = a;
        liczbaPokoi = p;
        powierzchnia = pow;
    }

    // Metoda do wyświetlania informacji
    void wyswietlInfo() {
        std::cout << "Adres: " << adres << ", Liczba pokoi: " << liczbaPokoi
            << ", Powierzchnia: " << powierzchnia << " m²" << std::endl;
    }

    // Metoda sprawdzająca, czy dom jest duży
    bool jestDuzy() {
        return (powierzchnia > 100.0);
    }
};

int main() {
    Dom mojDom("Ul. Główna 123, Warszawa", 4, 120.5); // Tworzenie obiektu
    mojDom.wyswietlInfo(); // Wyświetlenie info

    if (mojDom.jestDuzy()) {
        std::cout << "Dom jest duży." << std::endl;
    } else {
        std::cout << "Dom jest mały." << std::endl;
    }
    return 0;
}
```

