Lekcja 4

Konstruktory i inicjalizacja obiektów

Jak powstaje obiekt i skąd bierze swoje pierwsze wartości?

Klasa opisuje plan obiektu

Konstruktor uruchamia się przy tworzeniu obiektu

Inicjalizacja ustawia wartości początkowe

Cel lekcji: uczeń rozumie po co stosuje się konstruktor i potrafi przewidzieć, jakie wartości otrzyma obiekt po utworzeniu.

Po co nam konstruktor?

Najprościej: żeby obiekt od początku miał sensowny stan.

- Obiekt to konkretny egzemplarz klasy, np. jeden uczeń, jeden samochód, jeden telefon.
- Po utworzeniu obiektu pola powinny mieć wartości, które da się logicznie wykorzystać.
- Konstruktor pozwala nadać te wartości od razu, w kontrolowany sposób.

Bez konstruktora: obiekt może być „pusty” albo niepełny.

Z konstruktorem: obiekt dostaje dane startowe w chwili tworzenia.

Myśl dla klasy: konstruktor działa jak formularz przy zakładaniu konta — zanim konto powstanie, podajesz podstawowe dane.

Przypomnienie: klasa, obiekt i pola

Konstruktor ma sens dopiero wtedy, gdy pamiętamy, czym jest obiekt.

```
class Uczen {
public:
    string imie;      // pole: przechowuje imię
ucznia
    int wiek;         // pole: przechowuje wiek
ucznia

    void przedstawSie() {
        // metoda: zachowanie obiektu
        cout << "Mam na imie " << imie;
    }
};
```

Klasa = projekt / przepis / szablon

Obiekt = konkretny egzemplarz tej klasy

Pole = informacja zapisana w obiekcie

Metoda = czynność, którą obiekt potrafi wykonać

Na tym slajdzie jeszcze nie ma konstruktora. To tylko baza, do której za chwilę dodamy mechanizm tworzenia obiektu z danymi początkowymi.

Problem: obiekt bez wartości początkowych

Uczniowie powinni zobaczyć, że samo stworzenie obiektu to za mało.

```
Uczen u1;                 // tworzymy obiekt klasy
Uczen

u1.imie = "Ala";        // dopiero teraz
wpisujemy imię
u1.wiek = 17;           // dopiero teraz
wpisujemy wiek

u1.przedstawSie();      // korzystamy z danych
obiektu
```

Kod działa, ale ma słaby punkt:

- najpierw powstaje obiekt,
- dopiero później ktoś ręcznie ustawia pola,
- łatwo zapomnieć o jednym polu,
- łatwo ustawić dane w złej kolejności.

Konstruktor rozwiązuje ten problem: wymusza ustawienie danych przy tworzeniu obiektu.

Definicja konstruktora

To specjalna metoda, ale nie wygląda dokładnie jak zwykła metoda.

Konstruktor to specjalny fragment kodu w klasie, który uruchamia się automatycznie w momencie tworzenia obiektu.

```
class Uczen {
public:
    string imie;
    int wiek;

    Uczen() {
        // to jest konstruktor
    }
};
```

- Konstruktor ma taką samą nazwę jak klasa.
- Nie ma typu zwracanego, nawet void.
- Może ustawiać pola obiektu.
- Może przyjmować parametry, czyli dane potrzebne do utworzenia obiektu.

Do zapamiętania: jeśli klasa nazywa się Uczen, konstruktor też musi nazywać się Uczen().

Konstruktor domyślny

Domyślny, czyli taki, który nie wymaga żadnych danych w nawiasach.

```
class Telefon {
public:
    string marka;
    int bateria;

    Telefon() {
        // konstruktor domyślny
        marka = "Nieznana";  // wartość początkowa
        bateria = 100;       // startowo 100%
baterii
    }
};

Telefon t1; // uruchamia się Telefon()
```

Co się dzieje?

- Tworzymy obiekt t1.
- C++ automatycznie uruchamia konstruktor Telefon().
- Pola marka i bateria dostają wartości startowe.
- Nie musimy wpisywać tych danych ręcznie po utworzeniu obiektu.

To dobry przykład, gdy chcemy mieć bezpieczne wartości domyślne.

Konstruktor z parametrami

Parametry pozwalają przekazać dane do obiektu w chwili jego tworzenia.

```
class Uczen {
public:
    string imie;
    int wiek;

    Uczen(string noweImie, int nowyWiek) {
        // parametry są danymi z zewnątrz
        imie = noweImie;   // pole imie dostaje wartość
parametru
        wiek = nowyWiek;   // pole wiek dostaje wartość
parametru
    }
};

Uczen u1("Ala", 17); // tworzymy obiekt z danymi
```

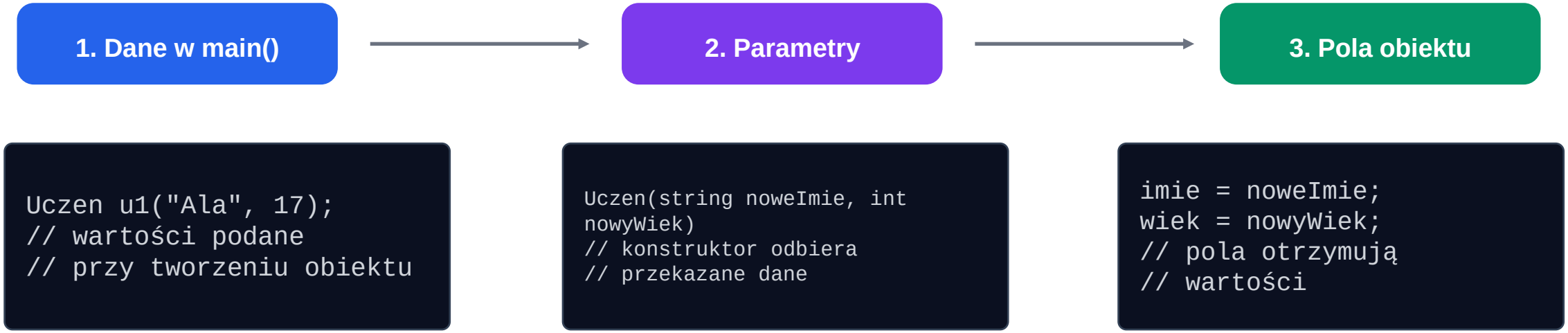
Jak to czytać?

- "Ala" trafia do parametru noweImie.
- 17 trafia do parametru nowyWiek.
- W konstruktorze wartości są przypisane do pól obiektu.
- Po tej instrukcji obiekt u1 jest gotowy do użycia.

To najczęstszy i najważniejszy typ konstruktora na tej lekcji.

Wizualnie: skąd dokąd płyną dane?

Ten slajd można omówić powoli, pokazując kolejność kroków.



Wniosek: konstruktor jest łącznikiem między danymi podanymi przy tworzeniu obiektu a polami zapisanymi wewnątrz obiektu.

Przykład: Samochód

Dobry przykład, bo uczniowie intuicyjnie rozumieją markę, model i rok produkcji.

```
class Samochod {
public:
    string marka;
    string model;
    int rok;

    Samochod(string m, string mod, int r) {
        // m, mod i r to krótkie nazwy parametrów
        marka = m;     // pole marka dostaje przekazaną markę
        model = mod;   // pole model dostaje przekazany model
        rok = r;       // pole rok dostaje przekazany rok
    }

    void pokaz() {
        cout << marka << " " << model << " " << rok;
    }
};
```

Co tu jest ważne?

- Pola opisują stan samochodu.
- Konstruktor wymaga trzech informacji.
- Bez tych informacji trudno mówić o konkretnym samochodzie.
- Metoda pokaz() wypisuje gotowy opis obiektu.

Pytanie do klasy: które dane są obowiązkowe, a które można by ustawić domyślnie?

Tworzenie obiektów z konstruktorem

Jedna klasa, wiele obiektów, różne wartości.

```
Samochod s1("Toyota", "Corolla", 2020);
Samochod s2("Ford", "Focus", 2018);
Samochod s3("Kia", "Ceed", 2022);

// Każda linia tworzy inny obiekt.
// Konstruktor uruchamia się osobno dla s1, s2 i s3.

s1.pokaz(); // Toyota Corolla 2020
```

To nie są trzy klasy.

To są trzy obiekty tej samej klasy.

Każdy ma własne wartości pól.

Warto mocno podkreślić uczniom: klasa jest jedna, ale obiektów może być bardzo dużo.

Konstruktor a zwykła metoda

Uczniowie często mylą konstruktor z metodą — ten slajd porządkuje różnicę.

Cecha	Konstruktor	Zwykła metoda
Kiedy działa?	Automatycznie przy tworzeniu obiektu	Gdy ją wywołamy, np. obiekt.metoda()
Nazwa	Taka sama jak nazwa klasy	Dowolna sensowna nazwa
Typ zwracany	Nie ma typu zwracanego	Może mieć void, int, string itd.
Zadanie	Ustawić stan początkowy obiektu	Wykonać zachowanie obiektu

Proste hasło: konstruktor przygotowuje obiekt do życia, metoda każe mu coś zrobić.

Częsty błąd: typ zwracany przy konstruktorze

W C++ konstruktor nie może mieć typu zwracanego.

```
class Uczen {
public:
    void Uczen() {
        // BŁĄD: to nie jest konstruktor
    }
};
```

Dlaczego to błąd?

- Konstruktor nie ma typu zwracanego.
- Nie piszemy void przed nazwą konstruktora.
- Poprawny zapis to samo Uczen().

```
Uczen() {
    // POPRAWNIE: konstruktor klasy Uczen
}
```

Na sprawdzianie taki błąd jest bardzo łatwy do zauważenia.

Częsty błąd: brak danych przy tworzeniu obiektu

Jeśli konstruktor wymaga parametrów, trzeba je podać.

```
class Konto {
public:
    string login;
    int punkty;

    Konto(string l, int p) {
        login = l;
        punkty = p;
    }
};

Konto k1;              // BŁĄD: brakuje loginu i
punktów
Konto k2("admin", 10); // POPRAWNIE
```

Jak to wytłumaczyć uczniom?

- Konstruktor Konto(string l, int p) mówi: „potrzebuję dwóch danych”.
- Konto k1; nic nie podaje, więc obiekt nie ma jak powstać.
- Konto k2("admin", 10); przekazuje dokładnie to, czego wymaga konstruktor.

Reguła: liczba i typ argumentów muszą pasować do parametrów konstruktora.

Minićwiczenie 1: przewidź stan obiektu

Bez komputerów: uczniowie analizują kod wzrokiem i odpowiadają ustnie lub w zeszycie.

```
class Gra {
public:
    string tytul;
    int poziom;

    Gra(string t) {
        tytul = t;       // tytuł bierzemy z parametru
        poziom = 1;      // poziom zawsze startuje od 1
    }
};

Gra g1("Minecraft");
```

Pytania dla klasy:

- Jaką wartość ma g1.tytul?
- Jaką wartość ma g1.poziom?
- Która wartość przyszła z zewnątrz?
- Która wartość została ustawiona na stałe w konstruktorze?

Oczekiwana odpowiedź: tytul = "Minecraft", poziom = 1.

Minićwiczenie 2: zaprojektuj konstruktor

To zadanie można zrobić na tablicy wspólnie z klasą.

Wymaganie: klasa Produkt ma przechowywać nazwę produktu, cenę oraz liczbę sztuk w magazynie.

Krok 1
Jakie pola są potrzebne?

- nazwa
- cena
- ilosc

Krok 2
Jakie dane trzeba podać przy tworzeniu obiektu?

- nazwa produktu
- cena
- liczba sztuk

Krok 3
Co robi konstruktor?

Przepisuje dane z parametrów do pól obiektu.

```
Produkt(string n, double c, int i) {
    nazwa = n;    // pole nazwa dostaje parametr n
    cena = c;     // pole cena dostaje parametr c
    ilosc = i;    // pole ilosc dostaje parametr i
}
```

Podsumowanie lekcji

Najważniejsze pojęcia do zapamiętania przed kolejnym tematem.

Konstruktor
Specjalny kod uruchamiany przy tworzeniu obiektu.

Inicjalizacja
Nadanie polom wartości początkowych.

Parametry
Dane przekazywane do konstruktora.

- Konstruktor ma taką samą nazwę jak klasa i nie ma typu zwracanego.
- Konstruktor może być domyślny albo może przyjmować parametry.
- Dzięki konstruktorowi obiekt od razu ma poprawny stan.
- Przy tworzeniu obiektu trzeba podać dane zgodne z parametrami konstruktora.

Pytanie kontrolne: co się stanie, jeśli konstruktor wymaga dwóch parametrów, a przy tworzeniu obiektu nie podamy żadnego?