Szkoła Gospodarstwa Wiejskiego
Wydział Zastosowań Informatyki i Matematyki

PARADYGMATY PROGRAMOWANIA

Jednostka 1 – Wprowadzenie + Paradygmat imperatywny

Kierunek: Informatyka, semestr 5
Prowadzący: dr inż. Krzysztof Malczewski

1. Cel i zakres zajęć

Celem zajęć jest:

- zapoznanie studentów z ogólną klasyfikacją paradygmatów programowania,
- przypomnienie i utrwalenie podstaw **paradygmatu imperatywnego**,
- zrozumienie różnic między imperatywnym a innymi podejściami,
- praktyczne opanowanie podstawowych konstrukcji imperatywnych: zmiennych, przypisań, instrukcji sterujących, pętli i procedur,
- przygotowanie do późniejszych zajęć, w których kolejne paradygmaty będą porównywane i łączone.

Zakres tematyczny:

- historia i podstawy paradygmatów,
- imperatywne podejście do rozwiązywania problemów,
- podstawowe struktury sterujące i dane,
- implementacja prostych algorytmów krok po kroku.

---

2. Wprowadzenie teoretyczne

2.1 Czym jest paradygmat programowania?

Paradygmat programowania to **styl lub model myślenia o programach**, który determinuje:

- sposób opisu problemu,
- strukturę programu,
- mechanizmy sterowania wykonaniem,
- sposób reprezentacji danych.

Do najważniejszych paradygmatów należą:

- **imperatywny**,
- **funkcyjny**,
- **obiektowy**,
- **deklaratywny i logiczny**,
- **zdarzeniowy**,
- **współbieżny**,
- **aspektowy**,
- oraz paradygmaty hybrydowe.

2.2 Paradygmat imperatywny – definicja

Paradygmat imperatywny opisuje program jako **ciąg instrukcji**, które:

- **zmieniają stan pamięci programu**,
- są wykonywane jedna po drugiej, zgodnie z określonym porządkiem,
- mogą zawierać konstrukcje sterujące przepływem (warunki, pętle).

👉 Programista w tym paradygmacie **opisuje krok po kroku, *jak* coś wykonać**, a nie *co* ma być wynikiem (jak np. w deklaratywnym).

2.3 Podstawowe elementy paradygmatu imperatywnego

- **Zmienne i typy danych** – przechowują stan programu.
- **Instrukcje przypisania** – zmieniają stan zmiennych.
- **Instrukcje sterujące** – warunki (`if`, `switch`) i pętle (`for`, `while`).
- **Instrukcje wejścia/wyjścia** – interakcja z użytkownikiem.
- **Procedury / funkcje** – grupują kod w logiczne jednostki.
- **Sekwencyjność** – kolejność instrukcji ma znaczenie.

2.4 Przykład: Obliczenie sumy liczb od 1 do N

Imperatywnie: podajemy algorytm krok po kroku

1. Pobierz N
2. Ustaw zmienną `suma` na 0
3. Powtarzaj od 1 do N: dodaj bieżącą liczbę do `suma`
4. Wypisz `suma`

Python:

```
n = int(input("Podaj liczbę N: "))
suma = 0
for i in range(1, n+1):
    suma += i
print("Suma:", suma)
```

C:

```
#include <stdio.h>

int main() {
    int n, suma = 0;
    printf("Podaj liczbę N: ");
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) {
        suma += i;
```

```
    }
    printf("Suma: %d\n", suma);
    return 0;
}
```

---

3. Tutoriale krok po kroku

3.1 Tutorial 1 – Uruchomienie środowiska i pierwszy program

1. Wybierz język, w którym chcesz pracować (np. C lub Python).
2. Skonfiguruj środowisko:
    - C: edytor + kompilator GCC, lub IDE (Code::Blocks, Visual Studio).
    - Python: interpreter Python 3, edytor (np. VS Code, PyCharm).
3. Utwórz nowy plik `hello.c` lub `hello.py`.
4. Napisz program, który wypisuje:
5. `Witaj w świecie programowania imperatywnego!`
6. Uruchom i sprawdź działanie.

3.2 Tutorial 2 – Instrukcje warunkowe

Python:

```
x = int(input("Podaj liczbę: "))
if x > 0:
    print("Liczba dodatnia")
elif x == 0:
    print("Zero")
else:
    print("Liczba ujemna")
```

C:

```
int x;
printf("Podaj liczbę: ");
scanf("%d", &x);
if (x > 0) {
    printf("Liczba dodatnia\n");
} else if (x == 0) {
    printf("Zero\n");
} else {
    printf("Liczba ujemna\n");
}
```

✅ **Cel:** zrozumienie działania instrukcji warunkowych i wpływu warunku na przepływ programu.

3.3 Tutorial 3 – Pętle i instrukcje iteracyjne

Zadanie: oblicz silnię liczby N

Python:

```
n = int(input("Podaj N: "))
silnia = 1
for i in range(1, n+1):
    silnia *= i
print(f"{n}! = {silnia}")
```

C:

```
int n;
unsigned long long silnia = 1;
scanf("%d", &n);
for (int i = 1; i <= n; i++) {
    silnia *= i;
}
printf("%d! = %llu\n", n, silnia);
```

✅ **Cel:** zrozumienie działania pętli `for`, zmiennych pomocniczych i kolejności wykonywania instrukcji.

---

4. Zadania do samodzielnego wykonania

Zadanie 1 (obowiązkowe)

Napisz program, który:

- pobiera liczbę całkowitą N,
- oblicza sumę liczb parzystych od 1 do N,
- wypisuje wynik.

Zadanie 2 (obowiązkowe)

Napisz program, który:

- pobiera N,
- wypisuje wszystkie liczby od 1 do N w kolejności malejącej,
- użyj pętli `while`.

Zadanie 3 (dodatkowe)

Zaimplementuj **algorytm sortowania bąbelkowego** (Bubble Sort) dla tablicy liczb w języku wybranym przez Ciebie.
Pamiętaj o:

- wczytaniu danych,
- przechowywaniu w tablicy/listach,
- zamianach miejscami,
- wypisaniu wyniku.

---

5. Pytania kontrolne

1. Czym różni się paradygmat imperatywny od deklaratywnego?
2. Co oznacza „zmiana stanu programu”?
3. Dlaczego kolejność instrukcji ma znaczenie w imperatywnym stylu?
4. Jak działają instrukcje warunkowe i pętle?
5. Jakie konstrukcje językowe są kluczowe dla paradygmatu imperatywnego?

---

6. Checklisty

Przed oddaniem ćwiczenia sprawdź:

- Program kompiluje się i działa bez błędów.
- Zastosowano przynajmniej jedną instrukcję warunkową.
- W programie użyto pętli `for` lub `while`.
- Wyniki dla różnych danych wejściowych są poprawne.
- Kod jest czytelny i zawiera komentarze.

---

7. Wzór sprawozdania

Tytuł ćwiczenia: Jednostka 1 – Paradygmat imperatywny
Autor: .....
Grupa: .....
Data: .....

1. Cel ćwiczenia

(opis celu własnymi słowami)

2. Przebieg ćwiczenia

(opis wykonanych kroków, środowiska, programów)

3. Wyniki działania

(zrzuty ekranu / wyniki przykładowych danych wejściowych)

4. Wnioski

(krótkie podsumowanie, czego się nauczyłem i gdzie paradygmat imperatywny ma zastosowanie)

---

8. Aneks – Szablony i przykłady kodu

Szablon programu w C

```
#include <stdio.h>

int main() {
    // Twój kod tutaj
    return 0;
}
```

Szablon programu w Pythonie

```
def main():
    # Twój kod tutaj
    pass

if __name__ == "__main__":
    main()
```

Zakończenie

Po wykonaniu ćwiczenia student powinien:

- rozumieć podstawowe założenia paradygmatu imperatywnego,
- umieć zapisać prosty algorytm w postaci sekwencji instrukcji,
- znać konstrukcje sterujące i potrafić ich używać,
- potrafić przygotować i uruchomić prosty program w wybranym języku,
- przygotować poprawne sprawozdanie dokumentujące wykonane zadania.