4. Imperativní programování

1. Algoritmizace

- teoretické řešení určitého problému
- nezáleží na daném paradigmatu, ale imperativní má k algoritmizaci nejblíže
- vlastnosti:
  - elementárnost: jednoduchý soubor kroků
  - konečnost: počet elementárních kroků musí být konečný
  - determinovanost: pro vstupy stejné povahy, shodné výstupy
  - determinismus: každý krok jednoznačně a přesně definován
  - výstup: každý algoritmus musí mít alespoň jeden výstup

2. Imperativní paradigma

- popisuje výpočet pomocí přesně definovaného sledu příkazů (imperativ) a určuje tak postup (algoritmus), jak danou úlohu řešit
- Dělení:
  - **Naivní paradigma**: nesystematická syntaxe a sémantika, jazyk BASIC
  - **Nestrukturované paradigma**: velmi blízké assemblerům, lineární sekvence, která je narušována příkazy *GO TO*, což bylo nepraktické, COBOL či FORTRAN
  - **Strukturované paradigma**: vývojové stádium nestrukturovaných, přidané cykly, podmínky a strukturované funkce, vnáření instrukcí, C, Python, Pascal
- základní typy příkazů:
  - přiřazení: operace s informacema uložených v paměti
  - cykly: opakování příkazů několikrát
  - příkazy pro větvení: klasické podmínky, pokud splněno, program se vnoří
- určují, jak se daný problém bude řešit, instrukce za instrukcí, takže popisují *jak*, srovnej s deklarativním přístupem
- funkce vrací hodnotu, zatímco procedura nikoliv
- program je sadou proměnných, jež v závislosti na vyhodnocení podmínek mění pomocí příkazů svůj stav.
- základní metodou imperativního programování je procedurální programování, tyto termíny bývají proto často zaměňovány.

3. Prvky programu

Jazyk C

```c
// komentář
#include "stdio.h" // standardní knihovna pro vstup a výstup
#include "stdlib.h" // práce s pamětí

int main() { // hlavní funkce
    printf("Hello World!\n"); // vypsat na obrazovku, \n je konec řádku

    int fortytwo = 42; // celé číslo
    float pi = 3.14; // číslo s plovoucí čárkou

    printf("%d\n", pi); // %d je číslo s plovoucí čárkou

    int a = 5;
    int *b = &a; // b je ukazatel na a

    printf("%i is the address of %i\n", b, a); // %i specifikuje celé číslo

    printf("%i\n", *b); // dereference, jdi na adresu
    a = 10;
    printf("%i\n", *b); // b uchovává jenom adresu

    const float PI = 3.1415; // konstanty, nelze je měnit

    int array[5]; // Pole s fixní délkou

    int delka;
    scanf("%i", &delka); // načíst ze standardního vstupu

    // Práce s pamětí

    int *dynamicArray = (int*)malloc(delka); // definovat pole délky delka
    // pak je možné možné použít realloc
    dynamicArray = (int*)realloc(dynamicArray, delka + 10);

    for (int i = 0; i < 10; i++) { // smyčka
        if (i % 2) { // pokud číslo je sudé
            printf("the number is even\n");
        } else { // jinak
            printf("the number is odd\n");
        }
```

Figure 1: Ukázka programu a jeho prvků