

# Úvod. Programovací paradigmatata

## Programovací techniky

doc. Ing. Jiří Rybička, Dr.  
ústav informatiky  
PEF MENDELU v Brně  
rybicka@mendelu.cz

**Cíl: programování efektivně a bezpečně**

## **Cíl: programování efektivně a bezpečně**

- Obsah

## **Cíl: programování efektivně a bezpečně**

- Obsah
  - ① Rozšíření informací o implementačním jazyce

## Cíl: programování efektivně a bezpečně

- Obsah
  - 1 Rozšíření informací o implementačním jazyce
  - 2 Pojem abstraktního typu a jeho implementace

## Cíl: programování efektivně a bezpečně

- Obsah
  - ① Rozšíření informací o implementačním jazyce
  - ② Pojem abstraktního typu a jeho implementace
  - ③ Algoritmy, hodnocení složitosti

## Cíl: programování efektivně a bezpečně

- Obsah
  - ① Rozšíření informací o implementačním jazyce
  - ② Pojem abstraktního typu a jeho implementace
  - ③ Algoritmy, hodnocení složitosti
- Nástroje

## Cíl: programování efektivně a bezpečně

- Obsah
  - ① Rozšíření informací o implementačním jazyce
  - ② Pojem abstraktního typu a jeho implementace
  - ③ Algoritmy, hodnocení složitosti
- Nástroje
  - ① Výukový jazyk



## Cíl: programování efektivně a bezpečně

- Obsah
  - ① Rozšíření informací o implementačním jazyce
  - ② Pojem abstraktního typu a jeho implementace
  - ③ Algoritmy, hodnocení složitosti
- Nástroje
  - ① Výukový jazyk
  - ② Dávkové programy

## Cíl: programování efektivně a bezpečně

- Obsah
  - ① Rozšíření informací o implementačním jazyce
  - ② Pojem abstraktního typu a jeho implementace
  - ③ Algoritmy, hodnocení složitosti
- Nástroje
  - ① Výukový jazyk
  - ② Dávkové programy
- Hodnocení

## Cíl: programování efektivně a bezpečně

- Obsah
  - ① Rozšíření informací o implementačním jazyce
  - ② Pojem abstraktního typu a jeho implementace
  - ③ Algoritmy, hodnocení složitosti
- Nástroje
  - ① Výukový jazyk
  - ② Dávkové programy
- Hodnocení
  - ① Zkouška – příklady (nebo fragmenty)

## Cíl: programování efektivně a bezpečně

- Obsah
  - ➊ Rozšíření informací o implementačním jazyce
  - ➋ Pojem abstraktního typu a jeho implementace
  - ➌ Algoritmy, hodnocení složitosti
- Nástroje
  - ➊ Výukový jazyk
  - ➋ Dávkové programy
- Hodnocení
  - ➊ Zkouška – příklady (nebo fragmenty)
  - ➋ Uplatnění vhodného přístupu

## Cíl: programování efektivně a bezpečně

- Obsah
  - ➊ Rozšíření informací o implementačním jazyce
  - ➋ Pojem abstraktního typu a jeho implementace
  - ➌ Algoritmy, hodnocení složitosti
- Nástroje
  - ➊ Výukový jazyk
  - ➋ Dávkové programy
- Hodnocení
  - ➊ Zkouška – příklady (nebo fragmenty)
  - ➋ Uplatnění vhodného přístupu
  - ➌ Čas zpracování (70 minut)

# Programovací paradigmatata

- **Procedurální**

- **Procedurální**
  - Nejstarší a nejrozšířenější



- **Procedurální**
  - Nejstarší a nejrozšířenější
  - Odpovídá strojovému přístupu

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Procedurální**
  - Nejstarší a nejrozšířenější
  - Odpovídá strojovému přístupu
  - Popisuje krok za krokem řešení problému
  - Běžné programovací jazyky (Pascal, C, ...)
- **Funkcionální**

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Funkcionální**

- Vyčíslování funkcí

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Funkcionální**

- Vyčíslování funkcí
- Funkce a seznamy v parametrech funkcí

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Funkcionální**

- Vyčíslování funkcí
- Funkce a seznamy v parametrech funkcí
- LISP (tabulkové procesory)

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Funkcionální**

- Vyčíslování funkcí
- Funkce a seznamy v parametrech funkcí
- LISP (tabulkové procesory)

- **Logické**



- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Funkcionální**

- Vyčíslování funkcí
- Funkce a seznamy v parametrech funkcí
- LISP (tabulkové procesory)

- **Logické**

- Seznam faktů: axiomy, vztahy

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Funkcionální**

- Vyčíslování funkcí
- Funkce a seznamy v parametrech funkcí
- LISP (tabulkové procesory)

- **Logické**

- Seznam faktů: axiomy, vztahy
- Řešení dotazu

- **Procedurální**

- Nejstarší a nejrozšířenější
- Odpovídá strojovému přístupu
- Popisuje krok za krokem řešení problému
- Běžné programovací jazyky (Pascal, C, ...)

- **Funkcionální**

- Vyčíslování funkcí
- Funkce a seznamy v parametrech funkcí
- LISP (tabulkové procesory)

- **Logické**

- Seznam faktů: axiomy, vztahy
- Řešení dotazu
- ProLog



- Posloupnost instrukcí vyjádřená operačními kódy a absolutními adresami paměti

- Posloupnost instrukcí vyjádřená operačními kódy a absolutními adresami paměti
- Je nejbližší stroji, nejdálenější člověku; jediná forma, které procesor přímo rozumí

- Posloupnost instrukcí vyjádřená operačními kódy a absolutními adresami paměti
- Je nejbližší stroji, nejdál od člověka; jediná forma, které procesor přímo rozumí
- Dnes se prakticky nepoužívá přímo, je výsledkem překladu z jiné úrovně programovacího jazyka

- Posloupnost instrukcí vyjádřená operačními kódy a absolutními adresami paměti
- Je nejbližší stroji, nejdál od člověka; jediná forma, které procesor přímo rozumí
- Dnes se prakticky nepoužívá přímo, je výsledkem překladu z jiné úrovně programovacího jazyka
- Tvar neumožňuje efektivně provádět změny (nutné přepočítávat adresy kódu a proměnných v paměti)



# Jazyk symbolických instrukcí

- Posloupnost instrukcí vyjádřená symbolickými zkratkami (ADD, MUL, MOV), adresy v paměti mohou být pojmenovány identifikátory

# Jazyk symbolických instrukcí

- Posloupnost instrukcí vyjádřená symbolickými zkratkami (ADD, MUL, MOV), adresy v paměti mohou být pojmenovány identifikátory
- Detailní řízení činnosti stroje

- Posloupnost instrukcí vyjádřená symbolickými zkratkami (ADD, MUL, MOV), adresy v paměti mohou být pojmenovány identifikátory
- Detailní řízení činnosti stroje
- Používá se například pro programování ovladačů zařízení

- Posloupnost instrukcí vyjádřená symbolickými zkratkami (ADD, MUL, MOV), adresy v paměti mohou být pojmenovány identifikátory
- Detailní řízení činnosti stroje
- Používá se například pro programování ovladačů zařízení
- Překlad, linkování (spojování = assembly, assembler)

# Vyšší programovací jazyk

- Již ne instrukce, ale vyšší celky – příkazy

- Již ne instrukce, ale vyšší celky – příkazy
- Nezávislost na stroji a hardwarové architektuře



- Již ne instrukce, ale vyšší celky – příkazy
- Nezávislost na stroji a hardwarové architektuře
- Nástup strukturovaných metod, objektových metod

- Již ne instrukce, ale vyšší celky – příkazy
- Nezávislost na stroji a hardwarové architektuře
- Nástup strukturovaných metod, objektových metod
- Univerzalita jazyků (Fortran, C, Pascal)

- Již ne instrukce, ale vyšší celky – příkazy
- Nezávislost na stroji a hardwarové architektuře
- Nástup strukturovaných metod, objektových metod
- Univerzalita jazyků (Fortran, C, Pascal)
- Implementace jazyků – tvorba překladačů, zavlékání překladačů

# Jazyky 4. generace (4GL)

- Další ulehčení práce programátora

- Další ulehčení práce programátora
- Specializované aplikace (SQL)

- Další ulehčení práce programátora
- Specializované aplikace (SQL)
- Možná změna paradigmatu (ProLog)

- Další ulehčení práce programátora
- Specializované aplikace (SQL)
- Možná změna paradigmatu (ProLog)
- Koexistence s jazyky 3. generace v současné době



# Interpretační a generační překlad

- Kompilátor = překladač vyššího PJ do strojového kódu

# Interpretační a generační překlad

- Kompilátor = překladač vyššího PJ do strojového kódu
- Generační překladač tvoří spustitelný modul

# Interpretační a generační překlad

- Kompilátor = překladač vyššího PJ do strojového kódu
- Generační překladač tvoří spustitelný modul
- Vlastnosti: bohatá syntax, kontrola celého kódu, rychlý běh výsledku

# Interpretační a generační překlad

- Kompilátor = překladač vyššího PJ do strojového kódu
- Generační překladač tvoří spustitelný modul
- Vlastnosti: bohatá syntax, kontrola celého kódu, rychlý běh výsledku
- Interpretační překladač překládá a hned provádí každý příkaz (např. řádek)

# Interpretační a generační překlad

- Kompilátor = překladač vyššího PJ do strojového kódu
- Generační překladač tvoří spustitelný modul
- Vlastnosti: bohatá syntax, kontrola celého kódu, rychlý běh výsledku
- Interpretační překladač překládá a hned provádí každý příkaz (např. řádek)
- Vlastnosti: interaktivita, slabší kontrola, méně datových typů, pomalejší běh výsledku

# Dávkové a interaktivní aplikace

# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus



# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus
- Interaktivní aplikace musí řešit mnoho situací spojených se vstupem a s výstupem pro člověka

# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus
- Interaktivní aplikace musí řešit mnoho situací spojených se vstupem a s výstupem pro člověka
- Existují vývojové prostředky pro usnadnění návrhu a použití uživatelského rozhraní

# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus
- Interaktivní aplikace musí řešit mnoho situací spojených se vstupem a s výstupem pro člověka
- Existují vývojové prostředky pro usnadnění návrhu a použití uživatelského rozhraní
- Dávková aplikace – nemá uživatelské rozhraní

# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus
- Interaktivní aplikace musí řešit mnoho situací spojených se vstupem a s výstupem pro člověka
- Existují vývojové prostředky pro usnadnění návrhu a použití uživatelského rozhraní
- Dávková aplikace – nemá uživatelské rozhraní
- Efektivní a malé programy, komunikují přes příkazový řádek a standardní vstupy a výstupy

# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus
- Interaktivní aplikace musí řešit mnoho situací spojených se vstupem a s výstupem pro člověka
- Existují vývojové prostředky pro usnadnění návrhu a použití uživatelského rozhraní
- Dávková aplikace – nemá uživatelské rozhraní
- Efektivní a malé programy, komunikují přes příkazový řádek a standardní vstupy a výstupy
- Spojování efektivních a rychlých komponent v dávkách

# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus
- Interaktivní aplikace musí řešit mnoho situací spojených se vstupem a s výstupem pro člověka
- Existují vývojové prostředky pro usnadnění návrhu a použití uživatelského rozhraní
- Dávková aplikace – nemá uživatelské rozhraní
- Efektivní a malé programy, komunikují přes příkazový řádek a standardní vstupy a výstupy
- Spojování efektivních a rychlých komponent v dávkách
- Lze se soustředit pouze na algoritmus

# Dávkové a interaktivní aplikace

- Pravidlo 90 : 10 – 90 % kódu programu tvoří uživatelské rozhraní, zbytek je vlastní algoritmus
- Interaktivní aplikace musí řešit mnoho situací spojených se vstupem a s výstupem pro člověka
- Existují vývojové prostředky pro usnadnění návrhu a použití uživatelského rozhraní
- Dávková aplikace – nemá uživatelské rozhraní
- Efektivní a malé programy, komunikují přes příkazový řádek a standardní vstupy a výstupy
- Spojování efektivních a rychlých komponent v dávkách
- Lze se soustředit pouze na algoritmus
- Snadnější ladění při přípravě dat do vstupního souboru, snadnější diagnostika filtrováním výstupů