Úvod. Programovací paradigmata

Programovací techniky

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

Programovací paradigmata

Programovací paradigmata

- **Procedurální**

Programovací paradigmata

- **Procedurální**
  - Nejstarší a nejrozšířenější

Programovací paradigmata

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

Programovací paradigmata

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

Programovací paradigmata

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

Programovací paradigmata

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

Programovací paradigmata

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

Programovací paradigmata

- **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

Programovací paradigmata

- **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)

Programovací paradigmata

- **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**

Programovací paradigmata

- **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

Programovací paradigmata

- **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

Programovací paradigmata

- **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

Strojový kód

Strojový kód

- Posloupnost instrukcí vyjádřená operačními kódy a absolutními adresami paměti

Strojový kód

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

Strojový kód

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

Strojový kód

- Posloupnost instrukcí vyjádřená operačními kódy a absolutními adresami paměti
- Je nejbližší stroji, nejvzdálenější člověku; 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

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

Jazyk symbolických instrukcí

- 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

Jazyk symbolických instrukcí

- 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

Vyšší programovací jazyk

- Již ne instrukce, ale vyšší celky – příkazy

Vyšší programovací jazyk

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

Vyšší programovací jazyk

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

Vyšší programovací jazyk

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

Vyšší programovací jazyk

- 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)

Jazyky 4. generace (4GL)

- Další ulehčení práce programátora

Jazyky 4. generace (4GL)

- Další ulehčení práce programátora
- Specializované aplikace (SQL)

Jazyky 4. generace (4GL)

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

Jazyky 4. generace (4GL)

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

Interpretační a generační překlad

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