Otázka 21

Algoritmy a algoritmizace

Počítačové programy (neboli software) umožňují počítačům, aby přestaly být pouhou stavebnicí elektronických a jiných součástek a staly se pomocníkem v mnoha lidských činnostech.

Uživatelé počítače obvykle neovládají počítač přímo, ale využívají k tomu různé programy. (Příkladem lze uvést textový editor, operační systém nebo počítačovou hru.) Tento software musí být vytvořen jinými lidmi, ale ačkoli jsou programy určeny pro různé oblasti, existují společné základy, které se při tvorbě programů obvykle využívají. Právě na tyto základy je zaměřen tento kurz.

Tvorba programů je činností, které se většinou věnují vysoce specializovaní jednotlivci v rámci různých softwarových společností. Po absolvování výuky základů algoritmizace se proto pravděpodobně nestanete skvělými tvůrci programů, pokud však pochopíte principy, bude cíl výuky naplněn. V tom případě totiž budete mít předpoklady pro případné další (a již konkrétnější) studium této oblasti a ani základní pojmy programování vám již nebudou cizí.

Dnes již počítače pronikly téměř do všech oblastí lidské činnosti, používají se k řešení nejrůznějších úkolů. Postup, který je v počítači prováděn nějakým programem se nazývá **algoritmus (program)** a jeho tvorba **algoritmizace (programování)**.

Historie

Algoritmy vznikaly už dávno před zkonstruováním prvních počítačů. Slovo „algoritmus“ vzniklo ze jména perského matematika, který žil v 9. století a jmenoval se Mohammed al-Khowarizmí (v latinském přepise Algoritmus). Zabýval se především pravidly pro aritmetické operace s čísly. Například Eukleidův algoritmus pro výpočet největšího společného dělitele dvou přirozených čísel pochází už z 4. až 3. století před naším letopočtem.

Algoritmus

Algoritmus je přesný popis, popisující určitý proces, který vede od měnitelných vstupních údajů k požadovaným výsledkům.
Jinými slovy – algoritmus je jednoznačný a přesný popis řešení problému.

Použití:

- při konstrukci či analýze algoritmů
- pro dokumentaci, pro zachycení myšlenky

Účel:

- názornost a přehlednost pro pochopení při algoritmizaci
- jednoznačný převod z textu programu do vývojového diagramu a naopak
- spíše nevhodné při návrhu celého (složitějšího) programu

Vstupní údaje

- informace, ze kterých při řešení úlohy vycházíme, musí splňovat vstupní podmínku

Výstupní údaje

- nově získané informace, které jsou výsledkem realizace algoritmu, musí splňovat výstupní podmínku

Každý algoritmus musí mít tyto vlastnosti:

- **správnost:** výsledek, který vznikne použitím algoritmu, musí být správný
- **resultativnost:** po konečném počtu kroků dospěje k řešení (vrátí třeba jen chybové hlášení)
- **konečnost:** algoritmus se nezacyklí, po určitém počtu kroků skončí
- **determinovanost:** v každém kroku je jednoznačně určen způsob pokračování práce algoritmu
- **hromadnost:** znamená, že algoritmus lze použít pro řešení obecné úlohy, tj. že nepopisujeme postup jedné úlohy, ale poslouží k řešení libovolné úlohy, která patří do jisté třídy úloh
- **opakovatelnost:** algoritmus vede vždy ke stejným výsledkům, jsou-li zadána stejná data

Algoritmus nemusí nutně vyžadovat vstupní údaje a vracet výstupní.

Některé problémy lze řešit více způsoby – různými algoritmy, které se mohou svým postupem značně lišit, ale vždy musí vést ke stejným výsledkům.

Naší snahou je vybrat pro řešení problému co nejefektivnější algoritmus, který řeší problém v co nejkratším čase, je přehledný a srozumitelný.

Algoritmus lze vyjádřit:

- **slovně:** jednotlivé kroky postupu jsou vyjádřeny větami v přirozeném jazyce
- **graficky:** jednotlivé kroky jsou popsány grafickými značkami se slovním popisem, například pomocí tzv. vývojových diagramů
- **matematicky:** soustavou rovnic, vztahem mezi veličinami
- **programem:** jednotlivé kroky jsou popsány instrukcemi určitého procesoru

Programování = zakódování algoritmu do zvoleného programovacího jazyka

Algoritmizace = proces vytváření a sestavování algoritmů

Efektivnost algoritmu

danou úlohu řeší více algoritmů, vybíráme efektivnější podle určitých kritérií:

- **časové:** úloha vyřešena v kratším čase (uvažujeme strojový čas tj. počet instrukcí procesoru)
- **paměťové:** spotřeba paměti
- **přehlednost, srozumitelnost:** (důležité pro další vývoj a úpravy)

Programovací jazyk = umělý jazyk jenž se používá pro definování sekvence programových příkazů, které lze zpracovat na počítači. Algoritmus má obecnou povahu, zatímco implementace algoritmu v určitém programovacím jazyku je ryze konkrétní.

Algoritmizace

Algoritmizace je přesný postup, který se používá při tvorbě programu pro počítač, jehož prostřednictvím lze řešit nějaký konkrétní problém.

Algoritmizaci lze rozdělit do několika kroků:

1. Formulace problému
2. Analýza úlohy
3. Vytvoření algoritmu
4. Sestavení programu
5. Odladění programu

Formulace problému

V této etapě je třeba přesně formulovat požadavky, určit výchozí hodnoty, požadované výsledky, jejich formu a přesnost řešení. Tvůrce algoritmu musí dokonale rozumět řešenému problému, jinak nemůže algoritmus sestavit – v praxi programátoři spolupracují s odborníky z oblastí, pro které mají vytvořit algoritmus.

Analýza úlohy

Při analýze úlohy si ověříme, zda je úloha řešitelná a uděláme si první představu o jejím řešení. Dále zjistíme, zda výchozí hodnoty jsou k řešení postačující a zda má úloha více řešení. Podle charakteru úlohy vybereme nejvhodnější a pokud možno nejjednodušší řešení.

Vytvoření algoritmu úlohy

Sestavíme jednoznačný sled jednotlivých instrukcí (příkazů), které je třeba provést, aby byl úkol správně vyřešen. Algoritmus přesně popisuje postup zpracování daného úkolu, nedává však odpověď na daný problém, ale pouze postup, jak ji získat.

Sestavení programu

Na základě algoritmu řešené úlohy sestavíme program (zdrojový text) v konkrétním programovacím jazyce. Ze zdrojového textu se pomocí překladače do strojového kódu vytvoří spustitelný program (případně interpretem se přeloží a spustí jednotlivé příkazy programu). Lze říci, že dobře provedená analýza úlohy a algoritmizace je velmi důležitá pro řešení daného problému a je základním předpokladem sestavení programu pro počítač.

Odladění programu

Odladěním chceme odstranit chyby z programu. Nejčastější chyby jsou chyby v zápise, tzv. **syntaktické** – ty odhalí překladač a dělají je i zkušení programátoři.
Horší jsou **logické** chyby, které vyplývají z nesprávně navrženého algoritmu, nebo chyby, které vzniknou špatným předpokladem v etapě formulace nebo analýzy úlohy – projeví se nesprávnou

činností programu nebo špatnými výsledky – při odstraňování těchto chyb může pomoci ladící program *(debugger)* umožňující sledování aktuálního stavu proměnných a krokování.
Teprve po odstranění všech druhů chyb můžeme program použít k praktickému řešení úloh.

Skutečný postup při algoritmizaci v praxi

Proces psaní příkazů v programovacím jazyce se nazývá **programování**. Aby programování skutečně vedlo k požadovanému výsledku, podílí se na vytvoření programu **obvykle více osob**.

V prvé řadě je třeba přesně určit, co bude program umět – o tuto činnost se obvykle stará **analytik**. Analytik poté vypracuje určité zadání (nejprve zjistí a popíše řešený problém a poté naznačí jeho řešení), které předá **vývojáři** (označovanému také jako **programátor**). Vývojář podle popisu vytvoří program (přepíše řešení do programovacího jazyka). Zápis programu v programovacím jazyce se nazývá **zdrojový kód**. Poté je spuštěn program, který dokáže přeložit tento zdrojový kód do jazyka počítače, a vznikne tak spustitelný program. Prvotní otestování programu se nazývá jeho **ladění**.

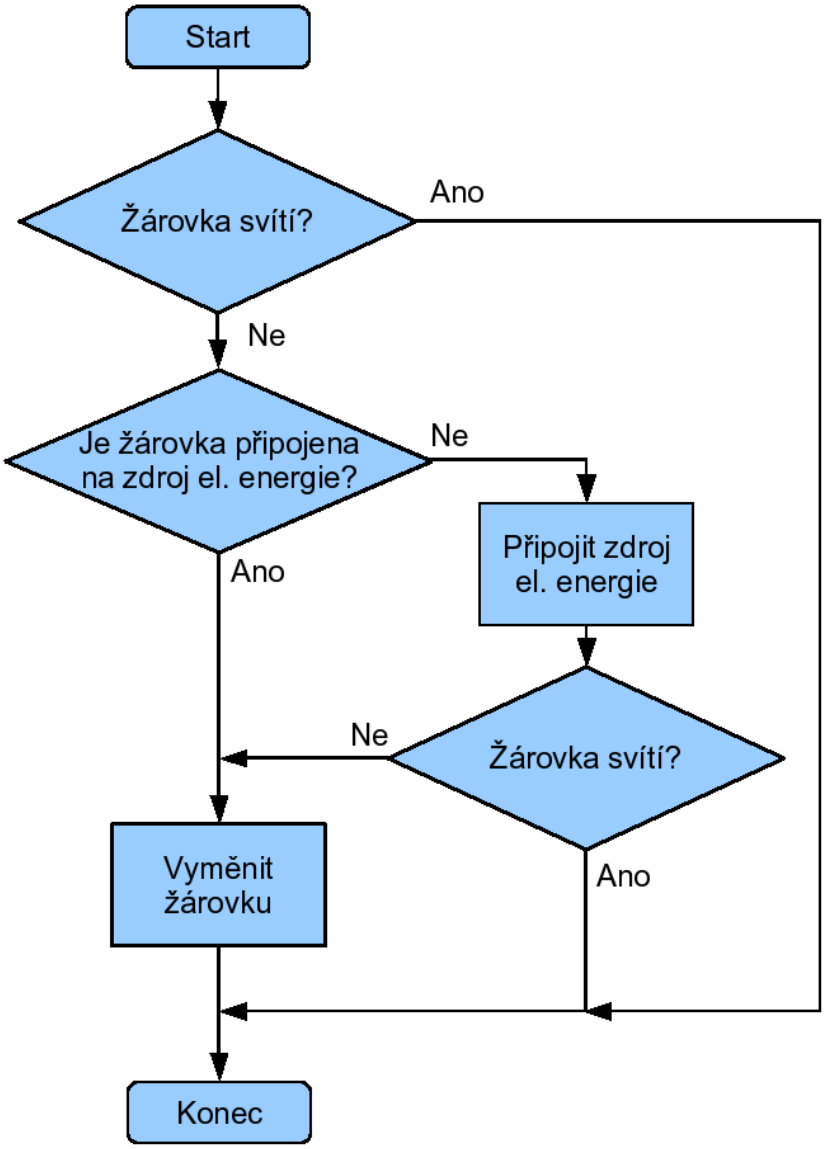
Vytvořením spustitelného programu práce ani zdaleka nekončí. Tento program dostane **tester**, který mu zadává různá data a zkoumá chování programu. Pokud objeví chybu, pak ji předá zpět programátorovi k dořešení.

Poté je program předán uživatelům. V průběhu práce s programem může docházet k určitým požadavkům na změny – například v důsledku nakoupení nového počítače nebo pro vytvoření další funkce, kterou původní program neobsahoval. Stejně tak mohou uživatelé chtít, aby jim bylo vysvětleno, jak mají software používat. Této části „života programu“ se říká **údržba a podpora**.

Vývojové diagramy

Jedním z mnoha způsobů znázornění algoritmů jsou vývojové diagramy.
Ve vývojových diagramech se používá několik typů značek, z nichž každé je přiřazen určitý význam.

Do těchto značek se vpisují operace nebo skupiny operací, které se mají provést.



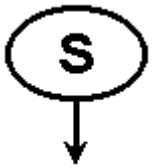
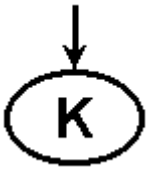
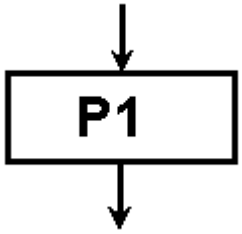
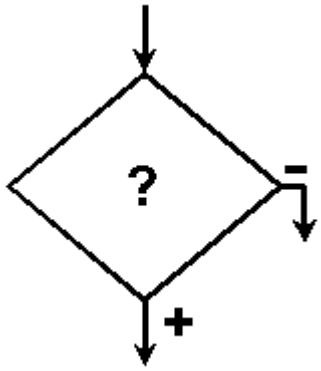
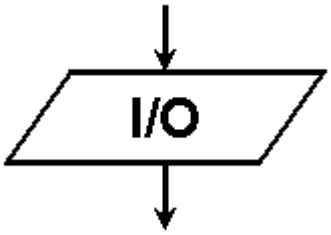
Vývojový diagram je druh **diagramu**, který slouží k grafickému znázornění jednotlivých kroků **algoritmu** nebo obecného procesu. Je to grafické znázornění logické struktury řešeného úkolu. Vývojový diagram používá pro znázornění jednotlivých kroků algoritmu symboly, které jsou navzájem propojeny pomocí orientovaných šipek. Symboly reprezentují jednotlivé procesy, šipky tok řízení.

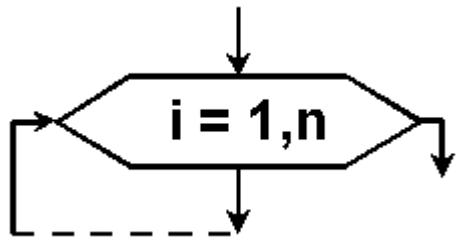
Již bylo řečeno, že před začátkem programování je vhodné nejprve přesně popsat nejen co řešíme, ale také jak to řešit (jedna z činností **zejména koho?**). Velmi srozumitelnou možností jak popsat, co bude počítač provádět, představuje vývojový diagram.

Velmi častým problémem začínajících programátorů je právě skutečnost, že si nejprve dobře nerozmyslí, co vlastně chtějí řešit ani jak přesně bude program pracovat. Pokud je problém složitý, je velmi obtížné uvědomit si všechny souvislosti bez přesného zadání úlohy a kostry postupu řešení.

Vývojový diagram je grafickým vyjádřením algoritmu. Jedná se o posloupnost geometrických obrazců, které jsou propojeny spojnicemi. Které obrazce to mohou být, popisuje následující stránka.

Používané značky:

	začátek algoritmu
	konec algoritmu
	blok zpracování neboli příkazů (do bloku zapisujeme akce, které se mají provést např. přiřazovací příkazy)
	blok rozhodování (do bloku zapisujeme podmínku)
	blok vstupu nebo výstupu – zapisujeme sem příkazy vstupu nebo výstupu

	blok pro cyklus se známým počtem průchodů (opakování)
	spojka (pro rozsáhlé diagramy, rozdělené do několika částí)

Pomocí těchto značek se kreslí základní algoritmické konstrukce.