

Jan Tvrdík, 2010

ALGORITMIZACE A PROGRAMOVÁNÍ

DEFINICE POJMŮ

- × **Problém**
 - + nevyřešený, nežádoucí stav
 - + obvykle vyžaduje nějaké řešení
- × **Neřešitelný problém**
 - + problém, který není algoritmicky řešitelný
- × **Algoritmus**
 - + přesný návod či postup, kterým lze vyřešit daný problém
- × **Algoritmizace**
 - + proces vytváření a sestavování algoritmů

VLASTNOSTI ALGORITMŮ

- ✘ Determinovanost (předurčenost)
 - + algoritmus musí být přesný, srozumitelný a jednoznačný
- ✘ Opakovatelnost
 - + stejná vstupní data a podmínky vedou ke stále stejným výsledkům
- ✘ Hromadnost
 - + algoritmus slouží k řešení celé skupiny úloh, které se od sebe liší jen vstupními parametry (ty se mohou měnit v určitých mezích)
- ✘ Resultativnost (konečnost)
 - + hledané výsledky musíme získat po konečném počtu kroků
 - + algoritmus se nezacyklí

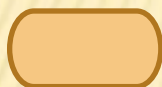
SMYSL TVORBY UMĚLÝCH JAZYKŮ

- × Pochopitelné jak pro lidi, tak pro počítače
- × Naučitelné
- × Jednoznačně definovaný význam slov a gramatická pravidla

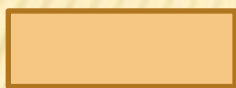
VÝVOJOVÉ DIAGRAMY

- × Slouží ke grafickému znázornění algoritmu
- × Jednotlivé dílčí operace znázorňuje symboly
- × Symboly se spojují pomocí úseček a šipek

BĚŽNĚ POUŽÍVANÉ SYMBOLY



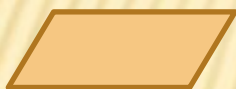
Začátek a konec programu



Příkaz



Větvení



Vstup nebo výstup



Cyklus s daným počtem opakování



Cyklus s podmínkou na začátku



Cyklus s podmínkou na konci

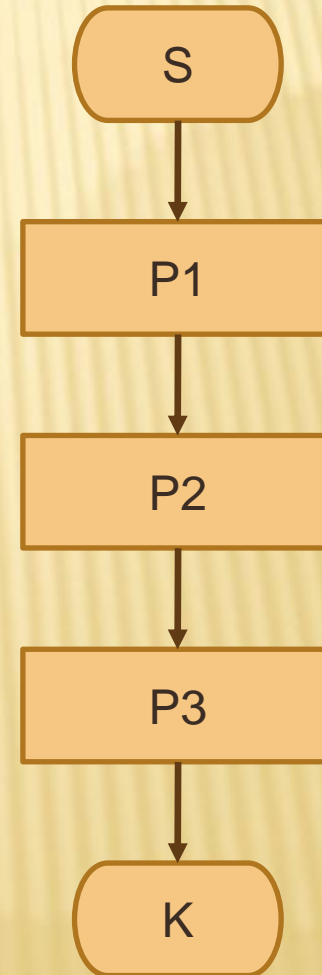


Nebo

SEKVENCE

- ✘ Posloupnost příkazů, které se postupně provedou
- ✘ Proved' příkazy P1, P2, P3!

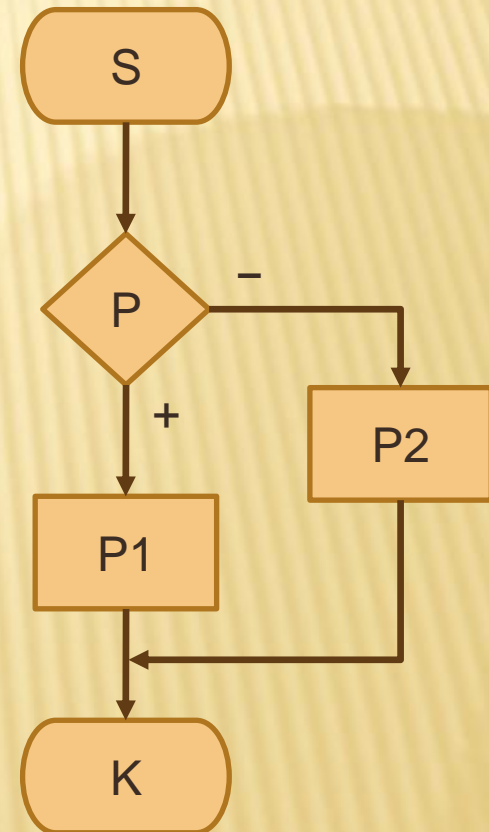
```
begin  
  P1;  
  P2;  
  P3;  
end;
```



VĚTVENÍ

- ✘ Rozdělí program do 2 větví podle toho, zda je nebo není splněna podmínka
- ✘ Jestliže platí podmínka P, proved' příkaz P1, jinak proved' příkaz P2!

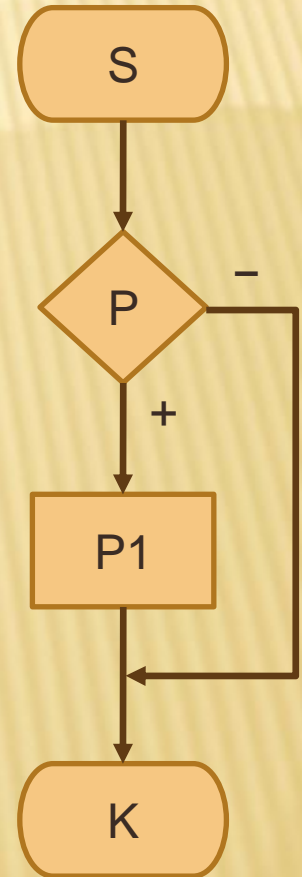
```
begin
  if P
    then P1;
    else P2;
end;
```



VĚTVENÍ BEZ ALTERNATIVY

- ✘ Provede příkaz pouze při splnění podmínky
- ✘ Jestliže platí podmínka P, proved' příkaz P1!

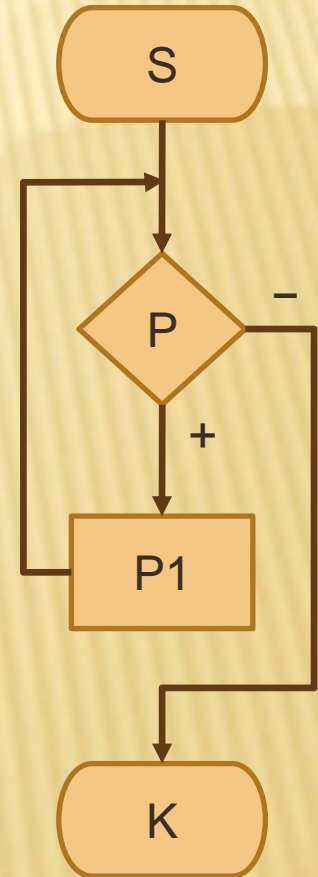
```
begin
  if P
    then P1;
end;
```



CYKLUS S PODMÍNKOU NA ZAČÁTKU

- ✘ Opakuje příkaz dokud platí daná podmínka
- ✘ Příkaz nemusí proběhnout ani jednou
- ✘ Dokud platí podmínka P, prováděj příkaz P1!

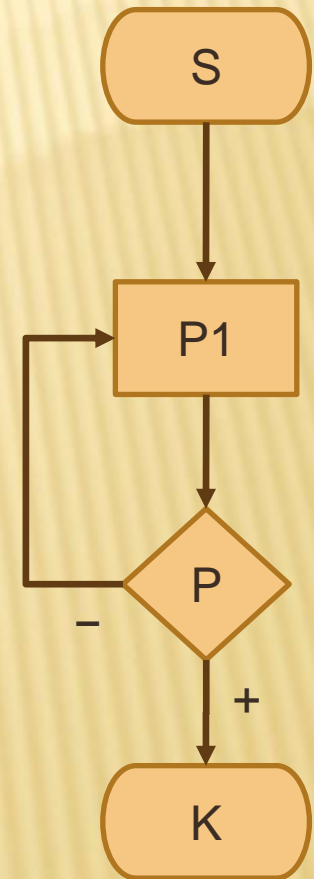
```
begin
  while P
    do P1;
end;
```



CYKLUS S PODMÍNKOU NA KONCI

- ✘ Opakuje příkaz, dokud není splněna daná podmínka.
- ✘ Příkaz proběhne alespoň jednou.
- ✘ Opakuj příkaz P1, až do splnění podmínky P!

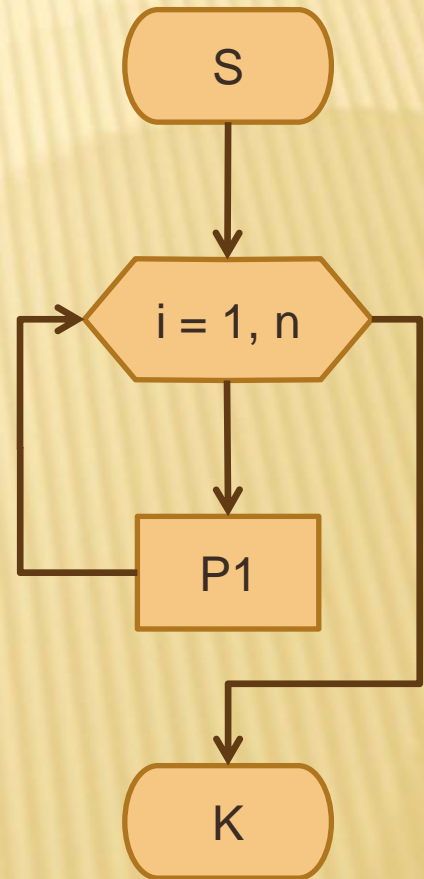
```
begin
  repeat
    P1;
  until P;
end;
```



CYKLUS S DANÝM POČTEM PRŮCHODŮ

- ✘ Provede $n \times$ daný příkaz
- ✘ Opakuj $n \times$ daný příkaz s parametrem i od 1 do n !

```
begin
  for i:=1 to n do
    P1;
end;
```



LOGICKÉ SPOJKY V PODMÍNKÁCH

- ✘ Logický součet = OR
 - + v matematice: disjunkce
 - + výraz je pravdivý, pokud je pravdivý alespoň jeden z operandů
- ✘ Logický součin = AND
 - + v matematice: konjunkce
 - + výraz je pravdivý, pokud jsou pravdivé oba operandy zároveň
- ✘ Negace = NOT
 - + v matematice: negace
 - + výraz je pravdivý, pokud operand není pravdivý

DĚLENÍ PROGRAMOVACÍCH JAZYKŮ

- ✘ Nižší programovací jazyky
 - + jejich instrukce odpovídají téměř přesně příkazům pro procesor
 - + závislé na architektuře procesoru
 - + Jazyk symbolických adres (assembly language), strojový kód
- ✘ Vyšší programovací jazyky
 - + abstrakce od architektury počítače a procesoru
 - + umožňují rychlejší vývoj aplikací
 - + tvoří většinu jazyků
 - + např. C#, Java, PHP, Pascal

DĚLENÍ PROGRAMOVACÍCH JAZYKŮ

× Kompilované jazyky

- + zdrojový kód se překládá do strojového kódu
- + rychlejší běh programu, horší přenositelnost na jinou platformu
- + např. C++, Pascal

× Interpretované jazyky

- + zdrojový kód je vykonáván prostřednictvím interpretu
- + pomalejší běh, nevyžaduje kompilaci
- + např. Unix shell, PHP, VBA

× Jazyky překládané do mezikódu

- + zdrojový kód je přeložen do mezikódu, který je následně vykonán pomocí interpretu
- + např. C#, Java

PŘEKLADAČ (KOMPILÁTOR)

- ✘ Slouží pro překlad algoritmů zapsaných ve vyšším programovacím jazyce do strojového kódu
- ✘ Typy překladačů
 - + Klasický – během překladu nelze zasahovat do jeho činnosti
 - + Konverzační (iterativní) – uživatel postupně zapisuje řádky zdrojového kódu
 - + Konverzační interpretační překladač – např. Logo, Karel
 - + Konverzační kompilační překladač

DATOVÝ TYP

- ✘ Určuje druh proměnné
- ✘ Má daný obor hodnot a operace, které s ním lze provádět
- ✘ Většinou jsou předdefinované v programovacím jazyce, lze ale i vytvářet nové (abstraktní datové typy)

ZÁKLADNÍ DATOVÉ TYPY

- × Prázdný (void)
- × Logická hodnota (boolean) – true nebo false
- × Celé číslo (integer)
- × Znak (char)
- × Reálné číslo (float, real)
- × Pole (array)
- × Text (string)
- × Výčtový typ (enum)

PROGRAMÁTORSKÁ CHYBA

- × Chyba, kterou udělal programátor při programování
- × Zranitelnost
 - + chyba způsobující bezpečnostní problém
- × Exploit
 - + program využívající zranitelnost
- × Bug
 - + anglický termín pro chybu
- × Debugování
 - + ladění
 - + proces odstraňování chyb

TYPY CHYB

× Syntaktické chyby

- + „pravopisné chyby“, porušení syntaxe daného jazyka
- + vznikají při kompilaci

× Logické chyby

- + chyby v logickém návrhu programu
- + nejhůře se hledají (program se bez problému spustí, nevypisuje žádné chybové hlášení, ale nepracuje správně)

× Běhové chyby

- + vznikají až při běhu programu

ODSTRAŇOVÁNÍ CHYB

- ✘ Metoda „rozděl a panuj“
 - + postupné zakomentování jednotlivých částí kódu
- ✘ Průběžné výpisy proměnných
- ✘ Debugger
 - + specializovaný program na vyhledávání chyb v jiných programech
 - + umožňuje krokovat kód
 - + umožňuje sledovat obsah proměnných
- ✘ Automatické testování
 - + zabraňuje návratu již opravených chyb
 - + snižuje výskyt chyb v budoucnu

PODPROGRAM

- ✘ Část programu vykonávající určitou činnost, která může být ovlivněna předávanými parametry
- ✘ Slouží k zpřehlednění programu a neopakování stejného kódu na více místech
- ✘ V Pascalu existují dva druhy – funkce a procedury
- ✘ Funkce na rozdíl od procedur vrací hodnotu

VIZUÁLNÍ PROGRAMOVÁNÍ

- ✘ Vizuální IDE umožňují uživateli vytvářet nové aplikace přemístěním programovacích stavebních bloků nebo uzlů a vytvořením vývojových diagramů nebo blokových schémat, které jsou dále přeloženy.

OBJEKTOVÉ PROGRAMOVÁNÍ

- ✘ Zkracováno jako OOP (Object-oriented programming)
- ✘ Snaží se modelovat řešení úlohy stejně jako v reálném světě
- ✘ Základem jsou objekty, které mají své vlastnosti a metody
 - + např. formulářové tlačítko, člověk, čtverec
- ✘ Koncepce
 - + Skládání – každý objekt může obsahovat jiné objekty
 - + Zapouzdření – každý objekt zpřístupňuje navenek rozhraní, pomocí kterého se s ním pracuje
 - + Dědičnost – objekty tvoří hierarchii, čímž od sebe navzájem přebírají vlastnosti
 - + Polymorfismus

ZDROJE

<http://www.spsemoh.cz/vyuka/algor/>

http://cmg.prostejov.cz/ivt/algo_prog.doc

<http://www.cmsps.cz/~marlib/diagramy/diagramy.html>

<http://cs.wikipedia.org/wiki/Problém>

<http://cs.wikipedia.org/wiki/Algoritmus>

http://cs.wikipedia.org/wiki/Programovací_jazyk

http://cs.wikipedia.org/wiki/Jazyk_symbolických_adres

http://cs.wikipedia.org/wiki/Vývojový_diagram

<http://cs.wikipedia.org/wiki/Překladač>

http://cs.wikipedia.org/wiki/Datový_typ

http://cs.wikipedia.org/wiki/Programátorská_chyba

http://cs.wikipedia.org/wiki/Vývojové_prostředí#Vizu.C3.A1n.C3.AD_programov.C3.A1n.C3.AD

http://cs.wikipedia.org/wiki/Objektové_programování

<http://www.cs.vsb.cz/kot/download/uti2007/uti-pr-07.pdf>

http://ncbr.chemi.muni.cz/~martinp/C2160/CChem_uvod.pdf

http://sportgym-strava.cz/ucitele/files/informatika/22_programovaci_jazyky.pdf

<https://akela.mendelu.cz/~mot/vyuka/skralq03.pdf>

<http://www.spsemoh.cz/vyuka/pascal/ppg.htm>