# Digitální učební materiál

| | |
|---|---|
| **Číslo projektu:** | **CZ.1.07/1.5.00/34.0548** |
| **Název školy:** | **Gymnázium, Trutnov, Jiráskovo náměstí 325** |
| **Název materiálu:** | **VY_32_INOVACE_142_IVT** |
| **Autor:** | **Ing. Pavel Bezděk** |
| **Tematický okruh:** | **Algoritmy** |
| **Datum tvorby:** | **červenec 2013** |
| **Ročník:** | **4. ročník a oktáva** |
| **Anotace:** | **Algoritmus II. – Správnost algoritmu** |
| **Metodický pokyn:** | **Při výuce nutno postupovat individuálně.<br>Části DUM – „ Pro hloubavé“ jsou určeny pro zájemce o studium na technických a matematicko-fyzikálních oborech vysokých škol.** |
| Pokud není uvedeno jinak, je použitý materiál z vlastních zdrojů autora DUM. | |

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

| | | | |
|---|---|---|---|
| **Autor** | **Ing. Pavel Bezděk** | | |
| **Vytvořeno dne** | **5. 7. 2013** | | |
| **Odpilotováno dne** | **18. 11. 2013** | **ve třídě** | **8.Y** |
| **Vzdělávací oblast** | **Informatika a informační a komunikační technologie** | | |
| **Vzdělávací obor** | **Informatika a výpočetní technika** | | |
| **Tematický okruh** | **Algoritmus** | | |
| **Téma** | **Algoritmus II . - Správnost algoritmu** | | |
| **Klíčová slova** | **Algoritmus, determinovanost, konečnost** | | |

# Algoritmus

## a jeho správnost

# Správnost algoritmu

**Algoritmus, který je správný, má tyto dvě vlastnosti:**

## 1. Konečnost

**Pro každá přípustná data algoritmus skončí v reálném čase**
**(nereálný čas – desítky, stovky, tisíce let)**

## 2. Parciální (částečná) správnost

**Jestliže algoritmus skončí, pak skončí se správným výsledkem**
**(Nemůže skončit se špatným výsledkem, když je správný !!!)**

# Eukleidův algoritmus *NSD největší společný dělitel*

gcd – *greatest common divisor*

Platí: **když X < Y** **NSD(X,Y) = NSD(X,Y-X)**
**když X > Y** **NSD(X,Y) = NSD(X-Y,Y)**
**když X = Y** **NSD(X,Y) = X**

Příklad: NSD(396,324) = NSD(72,324) = NSD(72,252) = NSD(72,180) = NSD(72,108) = NSD(72,36) = NSD(36,36) = 36

Algoritmus: dokud X ≠ Y od většího z čísel X, Y odečti menší z čísel X, Y

**Správnost (korektnost) Eukleidova algoritmu**

1. **Konečnost** - **na začátku výpočtu i stále v jeho průběhu je X>0, Y>0 - v každém kroku výpočtu se hodnota X+Y sníží alespoň o 1**
   → **nejpozději po X+Y krocích výpočet skončí, je tedy konečný.**

(Stále snižování s každým krokem hodnoty X+Y, indikuje konečnost. Nemůže se snižovat do nekonečna, nejpozději po X+Y krocích končí.)

**2. Parciální** (částečná) **správnost** - pro X = Y zjevně platí NSD(X,Y) = X, ukážeme, že pro X > Y platí NSD(X,Y) = NSD(X-Y,Y):
Nechť N=NSD(X,Y), tedy N dělí X a zároveň N dělí Y.
Proto také N dělí X-Y a je tedy N společným dělitelem X-Y a Y. Pokud by neplatilo, že N=NSD(X-Y,Y), musí existovat A>1 tak, že N.A=NSD(X-Y,Y).
Tedy N.A dělí X-Y i Y, takže N.A dělí i jejich součet, což je X. Jelikož N.A dělí Y a zároveň N.A dělí X, je N.A společným dělitelem X, Y, což je spor s tím, že N=NSD(X,Y). Proto N=NSD(X-Y,Y).

**Korektnost (správnost) algoritmu** si řeknete ještě jinak.
**Algoritmus je korektní, pokud pro každý vstup dá v konečném množství kroků správný výsledek.**

Pro dokazování korektnosti zavádíme dva pojmy – variant a invariant.

**Variant** je hodnota daná přirozeným číslem, která se v průběhu algoritmu stále snižuje, dokud nenabude hodnoty, při které algoritmus terminuje (končí). (Terminace: **algoritmus ukončí výpočet pro jakákoli přípustná vstupní data**.)
**U Eukleidova algoritmu je variant hodnota X+Y**

**Invariant** je naproti tomu **tvrzení, které platí na počátku algoritmu** (nebo po jeho prvním kroku), **platí i po provedení každého dalšího kroku a po terminaci (ukončení) algoritmu zaručuje správnost řešení**. (Formule parciální korektnosti: **tvrzení, které platí, pokud algoritmus svoji práci ukončí.)**

**U Eukleidova algoritmu je invariant tvrzení:**

**Když X<Y, platí NSD(X,Y)=NSD(X,Y-X).**
**Když X>Y, platí NSD(X,Y)=NSD(X-Y,Y)**
**Když X=Y, platí  NSD(X,Y)=X**

# Eukleidův algoritmus může mít 2 verze

## 1. Odčítací verze

Vstup: A,B
Opakuj dokud A<>B od většího z čísel A, B odečti menší a výsledek ulož tam, kde bylo větší číslo. Výstup: A (nebo B, je to jedno, neboť A=B).
Nebo můžeme říci, vyjdeme-li od dvojice čísel **A**,**B**. Vždy, když máme dvojici tvořenou různými čísly, vytvoř novou dvojici tak, že vezmeš menší z čísel staré dvojice a rozdíl těchto čísel. Pokud jsi tímto způsobem získal dvojici stejných čísel, pak jsou obě čísla v ní rovna hledanému největšímu společnému děliteli čísel **A** a **B**.
**Správnost algoritmu jsme již potvrdili.**

```
program NSD;
var A, B : integer;
begin
  read(A,B);
  while A<>B do
                    if A>B then A:=A-B
                           else B:=B-A;
  writeln(A);
 repeat until keypressed;
end.
```

## 2. Dělící verze (zbytek po dělení - modulo (mod))

Abychom (jako u předchozí verze) nemuseli opakovaně odečítat (pokud je např. A mnohem větší než B), **lze opakované odečítání nahradit zbytkem po celočíselném dělením (zbytek po celočíselném dělení bude výsledek, ke kterému bychom nakonec opakovaným odečítáním stejně došli).**

**Ale pozor, výpočet se v tomto případě zastaví, pokud A nebo B bude rovno 0 !!!!!**

### Správnost algoritmu:

**Variant** **X+Y (v našem případě A+B)**

**Invariant** **Když X>Y, platí NSD(X,Y)=NSD(X mod Y, Y);**
**Když X<Y, platí NSD(X,Y)=NSD(Y mod X, X);**

**Když (X=0 ∧ Y≠ 0), platí NSD(X,Y)=Y; Když (X ≠ 0 ∧ Y= 0), platí NSD(X,Y)=X;**

NSD(396,324) = NSD(72,324) = NSD(72,36) = NSD(0,36) = 36 (Je to rychlejší!)

```
program NSD2;
var A, B : integer;
begin
  read(A,B);
  while (A<>0)and(B<>0) do     if A>B  then A:=A mod B     else B:=B mod A;
  if A<>0   then writeln(A)  else writeln(B);
repeat until keypressed;
end.
```

## Program NSD odčítací verze Pascal

```
program NSD2;
uses Crt;
var A, B : integer;
begin
 write('Zadej 1. cislo: '); read(A);
 write('Zadej 2. cislo: '); read(B);
 while (A<>0)and(B<>0) do
              if A>B  then A:=A mod B
                    else B:=B mod A;
 if A<>0   then writeln('NSD = ',A)
         else writeln('NSD = ',B);
 repeat until keypressed;
end.
```

## Program NSD dělící verze Pascal

```
program NSD2;
uses Crt;
var A, B : integer;
begin
 write('Zadej 1. cislo: '); read(A);
 write('Zadej 2. cislo: '); read(B);
 while (A<>0)and(B<>0) do
              if A>B  then A:=A mod B
                    else B:=B mod A;
 if A<>0   then writeln('NSD = ',A)
         else writeln('NSD = ',B);
 repeat until keypressed;
end.
```

## Program NSD odčítací verze v C++

```
/* NSD - odcitaci metoda */
#include <iostream>
#include <math.h>          /* hlavickovy soubor matematicky */
using namespace std;
/* run this program using the console pauser or add your own getch, system("pause") or input loop */
int a, a1, b, b1;
int main(int argc, char** argv) {
cout<<("Nejvetsi spolecny delitel (znaceny NSD, ")<<endl;
cout<<(" prip. gcd z anglickeho greatest common divisor)")<<endl;
cout<<(" dvou celych cisel je nejvetsi cislo takove, ")<<endl;
cout<<("ze beze zbytku deli obe cisla, tzn. nejvetsi cislo,")<<endl;
cout<<("jimz jsou obe cisla delitelna.")<<endl;
cout<<(" Zadej prvni cislo:  ");
cin>>a;  cout<<endl;
cout<<(" Zadej druhe cislo:  ");
cin>>b;  cout<<endl;
a1=a; b1=b;
while (a!=b)
{ if (a>b) a=a-b;
     else b=b-a;
}
cout<<(" NSD (")<<a1<<", "<<b1<<" ) = "<<a;
                  return 0;
}
```

# Program NSD dělící verze v C++

```
/* NSD - delici metoda */
#include <iostream>
#include <math.h>          /* hlavickovy soubor matematicky */
using namespace std;
/* run this program using the console pauser or add your own getch, system("pause") or input loop */
int a, a1, b, b1;
int main(int argc, char** argv) {
cout<<("Nejvetsi spolecny delitel (znaceny NSD, ")<<endl;
cout<<(" prip. gcd z anglickeho greatest common divisor)")<<endl;
cout<<(" dvou celych cisel je nejvetsi cislo takove, ")<<endl;
cout<<("ze beze zbytku deli obe cisla, tzn. nejvetsi cislo,")<<endl;
cout<<("jimz jsou obe cisla delitelna.")<<endl;
cout<<(" Zadej prvni cislo:  ");
cin>>a;  cout<<endl;
cout<<(" Zadej druhe cislo:  ");
cin>>b;  cout<<endl;
a1=a; b1=b;
while ((a!=0) && (b!=0))
   { if (a>b) a=a % b;
     else b=b % a;
   }
     if (a!=0) {
                  cout<<(" NSD (")<<a1<<", "<<b1<<" ) = "<<a;
               }
      else cout<<(" NSD (")<<a1<<", "<<b1<<" ) = "<<b;
                return 0;
}
```

# Program Faktorial

```
Program Faktorial;
uses Crt;
var n,i: integer;     f,faktorialN:real;
begin
 writeln; writeln('Program Faktorial');  writeln;
 writeln('Maximalni hodnota: 33!');
 writeln('Hodnoty vetsi nez 33!, jsou moc velke pro rozsah proměnné real!');
 writeln('Po zadani hodnoty N stiskni enter!');  writeln;
 write('Zadej N: ');   readln(n);
 if n<=33 then     begin
                       writeln('Spravne zadana hodnota z mnoziny {1..33}');
                       writeln;    f:=1;
                       for i:=1 to n do f:=f*i;  faktorialN:=f;
                       writeln('Faktorial:  ',n,'! = ',faktorialN);
                   end

              else      writeln('Hodnota N mimo mnozinu {1..33}');
 writeln; writeln; writeln; writeln('Po precteni vysledku stiskni enter!');
 repeat until keypressed;
 end.
```

# Test

Posuďte, zda-li je  algoritmus programu Faktorial správný.

Zjistěte **variant** a **invariant** algoritmu programu Faktorial.

**Program faktoriál do 1754!**
**Pascal**

```
Program Faktorial;
uses Crt;
var n,i: integer;
    f,faktorialN: extended;    {10^(-4932)..10^4932}
begin
 writeln;  writeln;   writeln('Program Faktorial');   writeln;
writeln('Maximalni hodnota: 1754!');
 writeln('Hodnoty vetsi nez 1754!, jsou moc velke i pro pocitac!');
 writeln('Po zadani hodnoty N stiskni enter!');   writeln;   write('Zadej N: ');
readln(n);
 if n<=1754 then begin
              writeln('Spravne zadana hodnota z mnoziny {1..1754}');
              writeln;
              f:=1;   for i:=1 to n do f:=f*i;
              faktorialN:=f;
              writeln('Faktorial:  ',n,'! = ',faktorialN);
              end
            else writeln('Hodnota N mimo mnozinu {1..1754}');
 writeln;  writeln;  writeln;  writeln('Po precteni vysledku stiskni enter!');
 repeat until keypressed;
 end.
```

# Program faktoriál do 1754!
# C++

```
/* FAKTORIAL */
#include <iostream>        /* hlavickovy soubor vstupy a vostupy
*/
#include <math.h>          /* hlavickovy soubor matematicky */
using namespace std;

int  f,fak;
long double faktorial;
void faktor();

int main()
{

faktor();

            return 0;
}
```

```
void faktor()
{cout<<endl;
 cout<<("   Zadej hodnotu cisla ( maximalne cislo 1754
),")<<endl;
 cout<<("   pro ktere chces vypocitat faktorial: ");

cin >>fak; cout<<endl;
if (fak>1754){cout<<("  Byla prekrocena hodnota 1754!")<<endl;
            cout<<("  Vysledna hodnota faktorialu je tak velka, ze ji
PC uz nedokaze spocitat!!")<<endl;
            cout<<("  Cislo ma vice nez 4930 cislic!!!")<<endl;
            cout<<("  ")<<fak<<"! = chyba"<<endl;   }
else {
faktorial=1;
for  (f=1;f<=fak;f++)
   {faktorial=faktorial*f;
   }cout<<endl;
    cout<<("       Faktorial cisla:")<<endl;
    cout<<"       "<<fak<<"! = "<<faktorial<<endl;}
}
```

# Řešení testu

Variant: hodnota n-i {n, n-1, n-2, …….. 2, 1, 0}

Invariant: f:=1; {1!=1}

f:=f*i; {n!=(n-1)!*n}

# Test pro hloubavé

**Zjistěte jakou maximální hodnotu faktoriálu jsme schopni spočítat na počítači v Pascalu, budeme-li zvyšovat rozsah proměnných postupně od double až na extended .**

```
Program Faktorial;
uses Crt;
var n,i: integer;
   f,faktorialN: double;     {10^(-324) .. 10^308}
begin
 writeln;  writeln; writeln('Program Faktorial');
 writeln; writeln('Maximalni hodnota: 170! ');
 writeln('Po zadani hodnoty N stiskni enter!');
 writeln;  write('Zadej N: ');
 readln(n);
 if n<=170 then begin
                  writeln('Spravne zadana hodnota z mnoziny {1..170}');
                  writeln;
                  f:=1;
                  for i:=1 to n do f:=f*i;
                  faktorialN:=f;
                  writeln('Faktorial:  ',n,'! = ',faktorialN);
                      end
               else writeln('Hodnota N mimo mnozinu {1..170}');
 writeln;  writeln;  writeln;
 writeln('Po precteni vysledku stiskni enter!');
 repeat until keypressed;
end.
```

## Maximum je 1754!
## v Pascalu

```
Program Faktorial;
uses Crt;
var n,i: integer;
   f,faktorialN: extended;    {10^(-4932)..10^4932}
begin
 writeln;  writeln;   writeln('Program Faktorial');   writeln;   writeln('Maximalni
hodnota: 1754!');
 writeln('Hodnoty vetsi nez 1754!, jsou moc velke i pro pocitac!');
 writeln('Po zadani hodnoty N stiskni enter!');   writeln;   write('Zadej N: ');
readln(n);
 if n<=1754 then begin
             writeln('Spravne zadana hodnota z mnoziny {1..1754}');
             writeln;
             f:=1;
             for i:=1 to n do f:=f*i;
             faktorialN:=f;
             writeln('Faktorial:  ',n,'! = ',faktorialN);
             end
          else writeln('Hodnota N mimo mnozinu {1..1754}');
 writeln;  writeln;  writeln;  writeln('Po precteni vysledku stiskni enter!');
 repeat until keypressed;
 end.
```

# Program faktorial v C++

```
/* FAKTORIAL */
#include <iostream>        /* hlavickovy soubor vstupy a vostupy */
#include <math.h>          /* hlavickovy soubor matematicky */
using namespace std;
int  f,fak;
long double faktorial;
void faktor();
int main()
{faktor();
return 0;
}
void faktor()
{cout<<endl;
 cout<<("   Zadej hodnotu cisla ( maximalne cislo 1754 ),")<<endl;
 cout<<("   pro ktere chces vypocitat faktorial: ");
cin >>fak; cout<<endl;
if (fak>1754){cout<<("  Byla prekrocena hodnota 1754!")<<endl;
          cout<<("  Vysledna hodnota faktorialu je tak velka, ze ji PC uz nedokaze spocitat!!")<<endl;
          cout<<("  Cislo ma vice nez 4930 cislic!!!")<<endl;
          cout<<("  ")<<fak<<"! = chyba"<<endl;   }
else {
faktorial=1;
for  (f=1;f<=fak;f++)
  {faktorial=faktorial*f;
  }cout<<endl;
   cout<<("      Faktorial cisla:")<<endl;
   cout<<"      "<<fak<<"! = "<<faktorial<<endl;}
}
```

# Použité zdroje

Algoritmus. In: *Wikipedia: Otevřená encyklopedie* [online]. [cit. 2013-07-05]. Dostupné podle licence Creative Commons z www: http://cs.wikipedia.org/wiki/Algoritmus

BÖHM, Martin. *Programátorská kuchařka: Recepty z programátorské kuchařky* [online]. Praha: KSP MFF UK Praha, 2011/2012 [cit. 2013-07-05]. KSP, Korespondenční seminář z programování: Programátorské kuchařky, 24. ročník KSP. Dostupné z: http://ksp.mff.cuni.cz/tasks/24/cook1.html