

programy  
ke stažení na  
**WWW.GRADA.CZ**

# **Algoritmy v jazyku C a C++ praktický průvodce**

**Jiří Prokop**

- Seznámení s jazykem C a úvod do C++
- Vyhledávací algoritmy a algoritmy pro třídění
- Datové struktury a práce s grafy
- Soustavy lineárních rovnic a dynamické programování



## Upozornění pro čtenáře a uživatele této knihy

Všechna práva vyhrazena. Žádná část této tištěné či elektronické knihy nesmí být reprodukována a šířena v papírové, elektronické či jiné podobě bez předchozího písemného souhlasu nakladatele. Neoprávněné užití této knihy bude **trestně stíháno**.

*Používání elektronické verze knihy je umožněno jen osobě, která ji legálně nabyla a jen pro její osobní a vnitřní potřeby v rozsahu stanoveném autorským zákonem. Elektronická kniha je datový soubor, který lze užívat pouze v takové formě, v jaké jej lze stáhnout s portálu. Jakékoliv neoprávněné užití elektronické knihy nebo její části, spočívající např. v kopírování, úpravách, prodeji, pronajímání, půjčování, sdělování veřejnosti nebo jakémkoliv druhu obchodování nebo neobchodního šíření je zakázáno! Zejména je zakázána jakákoliv konverze datového souboru nebo extrakce části nebo celého textu, umisťování textu na servery, ze kterých je možno tento soubor dále stahovat, přitom není rozhodující, kdo takovéto sdílení umožnil. Je zakázáno sdělování údajů o uživatelském účtu jiným osobám, zasahování do technických prostředků, které chrání elektronickou knihu, případně omezují rozsah jejího užití. Uživatel také není oprávněn jakkoliv testovat, zkoušet či obcházet technické zabezpečení elektronické knihy.*



Copyright © Grada Publishing, a.s.



Copyright © Grada Publishing, a.s.

# Algoritmy v jazyku C a C++

## praktický průvodce

**Jiří Prokop**

Vydala Grada Publishing, a.s.  
U Průhonu 22, Praha 7  
jako svou 3473. publikaci

Odpovědný redaktor Zuzana Malečková  
Návrh vnitřního layoutu Miroslav Lochman  
Sazba Eva Grillová  
Návrh a grafická úprava obálky Vojtěch Kočí  
Počet stran 160  
První vydání, Praha 2009

© Grada Publishing, a.s., 2009  
Cover Photo © fotobanka allphoto

*V knize použité názvy programových produktů, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.*

Vytiskly Tiskárny Havlíčkův Brod, a.s.  
Husova ulice 1881, Havlíčkův Brod

ISBN 978-80-247-2751-6 (tištěná verze)  
ISBN 978-80-247-6622-5 (elektronická verze ve formátu PDF)  
© Grada Publishing, a.s. 2011

# Obsah

|           |   |
|-----------|---|
| Úvod..... | 9 |
|-----------|---|

|                 |    |
|-----------------|----|
| 1. Jazyk C..... | 11 |
|-----------------|----|

|                                   |    |
|-----------------------------------|----|
| 1.1 Stručný přehled jazyka C..... | 11 |
|-----------------------------------|----|

|                                               |    |
|-----------------------------------------------|----|
| 1.1.1 Deklarace .....                         | 11 |
| 1.1.2 Výrazy a přiřazení.....                 | 12 |
| 1.1.3 Priorita a asociativita operátorů ..... | 13 |
| 1.1.4 Příkazy a bloky .....                   | 14 |
| 1.1.5 Preprocesor .....                       | 15 |
| 1.1.6 Funkce .....                            | 16 |
| 1.1.7 Vstup a výstup.....                     | 17 |
| 1.1.8 Ukazatele .....                         | 19 |
| 1.1.9 Adresní aritmetika .....                | 20 |
| 1.1.10 Ukazatele a funkce.....                | 20 |
| 1.1.11 Pole .....                             | 20 |
| 1.1.12 Ukazatele a pole .....                 | 21 |
| 1.1.13 Řetězce znaků .....                    | 21 |
| 1.1.14 Vícerozměrná pole.....                 | 22 |

|                                |    |
|--------------------------------|----|
| 1.2 Jednoduché algoritmy ..... | 23 |
|--------------------------------|----|

|                                                                                            |    |
|--------------------------------------------------------------------------------------------|----|
| 1.2.1 Vyhledání minimálního prvku v neseřtříděném poli .....                               | 23 |
| 1.2.2 Vyhledání zadaného prvku v neseřtříděném poli .....                                  | 23 |
| 1.2.3 Určení hodnoty Ludolfova čísla pomocí rozvoje $\pi=4(1-1/3+1/5-1/7+1/9+\dots)$ ..... | 23 |
| 1.2.4 Mzdová výčetka .....                                                                 | 24 |
| 1.2.5 Největší společný dělitel dvou čísel .....                                           | 25 |
| 1.2.6 Pascalův trojúhelník .....                                                           | 25 |
| 1.2.7 Kalendář .....                                                                       | 26 |

|                  |    |
|------------------|----|
| 2. Rekurse ..... | 29 |
|------------------|----|

|                        |    |
|------------------------|----|
| 2.1 Hanojské věže..... | 30 |
|------------------------|----|

|                    |    |
|--------------------|----|
| 2.2 W-křivky ..... | 31 |
|--------------------|----|

|                                           |    |
|-------------------------------------------|----|
| 2.3 Fibonacciho čísla (posloupnost) ..... | 34 |
|-------------------------------------------|----|

|                                                     |           |
|-----------------------------------------------------|-----------|
| <b>3. Algoritmy pro třídění .....</b>               | <b>37</b> |
| 3.1 Třídění výběrem (selectsort).....               | 38        |
| 3.2 Třídění vkládáním (insertsort) .....            | 38        |
| 3.3 Bublínkové třídění (bubblesort).....            | 39        |
| 3.4 Časová a paměťová složitost.....                | 40        |
| 3.5 Třídění slučováním (mergesort).....             | 41        |
| 3.6 Třídění rozděláváním (quicksort).....           | 41        |
| 3.7 Shellův algoritmus .....                        | 42        |
| 3.8 Metoda „Rozděl a panuj“ .....                   | 43        |
| <b>4. Datové struktury.....</b>                     | <b>45</b> |
| 4.1 Dynamické datové struktury .....                | 46        |
| 4.1.1 Lineární spojový seznam.....                  | 46        |
| 4.1.2 Lineární spojový seznam seříděný .....        | 49        |
| 4.1.3 Seřídění vytvořeného lineárního seznamu ..... | 49        |
| 4.2 Zásobník a fronta.....                          | 53        |
| 4.3 Nerekurzivní verze quicksortu .....             | 55        |
| <b>5. Práce s grafy .....</b>                       | <b>57</b> |
| 5.1 Úvod do teorie grafů .....                      | 57        |
| 5.2 Topologické třídění .....                       | 59        |
| 5.3 Minimální kostra grafu.....                     | 61        |
| 5.4 Bipartitní graf .....                           | 63        |
| 5.5 Práce se soubory dat .....                      | 64        |
| 5.5.1 Datové proudy .....                           | 65        |
| 5.5.2 Proudý a vstup/výstup znaků .....             | 65        |
| 5.5.3 Proudý a vstup/výstup řetězců.....            | 66        |
| 5.5.4 Formátovaný vstup/výstup z/do proudu .....    | 66        |
| 5.5.5 Proudý a blokový přenos dat .....             | 66        |
| 5.5.6 Další užitečné funkce.....                    | 66        |

|     |                                                           |     |
|-----|-----------------------------------------------------------|-----|
| 5.6 | Vzdálenosti v grafu .....                                 | 68  |
| 6.  | Vyhledávací algoritmy .....                               | 73  |
| 6.1 | Binární hledání v seříděném poli .....                    | 73  |
| 6.2 | Binární vyhledávací strom .....                           | 74  |
| 6.3 | Vynechání vrcholu v binárním vyhledávacím<br>stromu ..... | 78  |
| 6.4 | Procházení stromem .....                                  | 84  |
| 6.5 | Transformace klíče .....                                  | 84  |
| 6.6 | Halda .....                                               | 85  |
| 6.7 | Využití haldy pro třídění – heapsort .....                | 87  |
| 7.  | Reprezentace aritmetického výrazu binárním stromem .....  | 91  |
| 7.1 | Vyhodnocení výrazu zadaného v postfixové notaci .....     | 92  |
| 7.2 | Převod infixové notace na postfixovou .....               | 94  |
| 7.3 | Převod postfixové notace na binární strom .....           | 97  |
| 8.  | Průchod stavovým prostorem .....                          | 101 |
| 8.1 | Prohledávání do šířky .....                               | 102 |
| 8.2 | Prohledávání s návratem (backtracking) .....              | 104 |
| 8.3 | Osm dam na šachovnici .....                               | 107 |
| 8.4 | Sudoku .....                                              | 109 |
| 8.5 | Hry pro 2 hráče .....                                     | 112 |
| 9.  | Úvod do C++ .....                                         | 115 |
| 9.1 | Nové možnosti jazyka .....                                | 115 |
| 9.2 | Objektové datové proudy .....                             | 116 |
| 9.3 | Objektově orientované programování .....                  | 116 |

|            |                                             |     |
|------------|---------------------------------------------|-----|
| 9.4        | Šablony .....                               | 119 |
| 10.        | Algoritmy numerické matematiky .....        | 123 |
| 10.1       | Řešení nelineární rovnice $f(x)=0$ .....    | 123 |
| 10.1.1     | Hornerovo schéma.....                       | 123 |
| 10.1.2     | Metoda půlení intervalu (bisekce) .....     | 124 |
| 10.1.3     | Metoda tětiv (regula falsi) .....           | 126 |
| 10.1.4     | Newtonova metoda (metoda tečen).....        | 127 |
| 10.2       | Interpolace .....                           | 130 |
| 10.2.1     | Newtonova interpolace .....                 | 130 |
| 10.2.2     | Lagrangeova interpolace.....                | 131 |
| 10.3       | Soustavy lineárních rovnic.....             | 132 |
| 10.3.1     | Gaussova eliminační metoda .....            | 132 |
| 10.3.2     | Iterační (Jacobiho) metoda.....             | 134 |
| 10.3.3     | Gauss-Seidelova metoda .....                | 136 |
| 11.        | Dynamické programování.....                 | 139 |
| 12.        | Vyhledání znakového řetězce v textu .....   | 143 |
| 12.1       | „Naivní“ algoritmus .....                   | 143 |
| 12.2       | Zjednodušený Boyer-Mooreův algoritmus ..... | 144 |
| 12.3       | Karp-Rabinův algoritmus .....               | 146 |
| Literatura | .....                                       | 149 |
| Rejstřík   | .....                                       | 151 |



# Úvod

V r. 2002, kdy jsem začal na Gymnáziu Christiana Dopplera vést seminář s názvem „Programování v jazyku C“, neexistovala na našem knižním trhu učebnice, která by se věnovala algoritmům a používala jazyk C. Algoritmy byly po řadu let prezentovány téměř výlučně v jazyku Pascal, např. [Wir89] a [Top95]. Musel jsem tedy během šesti let algoritmy pro účely výuky naprogramovat, a tak vznikl základ této knihy.

Kniha se nesnaží být učebnicí jazyka C, i když může být k užítku všem, kteří jazyk právě studují. Dobrých učebnic jazyka je dostatek, doporučit lze např. [Her04] nebo [Ka01], pro C++ [Vi02], [Vi97]. Jestliže jsem přesto zařadil do knihy alespoň stručný přehled jazyka C a také úvod do C++, je to proto, aby čtenář měl při studiu knihy vše potřebné pro porozumění zdrojovým textům algoritmů a nemusel hledat informace jinde.

Kdo je s jazykem C seznámen do té míry, že chápe nejdůležitější operátory, výrazy a přiřazení, příkazy pro řízení programu, příkazy vstupu a výstupu, funkce a vedle jednoduchých datových typů ještě pole, stačí mu to už ke studiu jednoduchých algoritmů. Takový přehled jazyka obsahuje právě první kapitola. Poté lze studovat kapitolu druhou, věnovanou rekurzi, a třetí, která se zabývá třídícími algoritmy. Teprve pro studium datových struktur je v kapitole 4 nutno rozšířit zatímní podmnožinu jazyka o struktury a dynamické přidělování paměti. Tyto znalosti jsou pak potřebné i pro pochopení algoritmů na grafech

a pro vyhledávání pomocí binárních stromů. Stromy se využívají také k reprezentaci aritmetických výrazů a pro počítačové řešení hlavolamů a her. Popsaná podmnožina jazyka je v těchto kapitolách dále rozšiřována podle potřeby. Algoritmy z kapitol 1 až 8 jsou napsány v jazyku C. Teprve 9. kapitola je úvodním popisem C++ a algoritmy v následujících kapitolách jsou v C++.

Z tohoto stručného průvodce obsahem knihy vyplývá samozřejmě doporučení studovat jednotlivé kapitoly postupně a bez přeskakování, protože v každé kapitole se počítá se znalostmi, které si čtenář přináší z kapitol předchozích. Dalším doporučením je studium aktivní. Usnadňuji ho tím, že všechny algoritmy rozdělené podle kapitol knihy lze najít na webových stránkách [www.grada.cz](http://www.grada.cz). Zdrojové texty tedy nemusí nikdo pracně vkládat, čtenář může provádět v programech úpravy, mnohde k tomu zdrojový text přímo vybízí tím, že části zdrojového textu jsou „ukryty“ v komentářích. Často lze algoritmus snáze pochopit, zobrazíme-li si některé mezivýsledky. Aktivní způsob studia je mimo jiné určitě mnohem zajímavější. Algoritmy jsou ověřeny s použitím kompilátoru Dev C++ a kompilátoru Microsoft Visual C++. Kdyby čtenáři měli ke knize jakékoli připomínky, mohou je sdělit na e-mailovou adresu [Jiri\\_Prokop@yahoo.com](mailto:Jiri_Prokop@yahoo.com). Přeji svým čtenářům mnoho úspěchů ve studiu.

# 1.

# Jazyk C

## 1.1 Stručný přehled jazyka C

Jazyk C rozlišuje velká a malá písmena. „Prog“, „prog“ a „PROG“ jsou tedy tři různé identifikátory. Identifikátory sestávají z písmen, číslic a podtržítka, číslice nesmí být na prvním místě. Pro oddělování klíčových slov, identifikátorů a konstant slouží oddělovače (tzv. „bílé znaky“). Všude tam, kde mohou být oddělovače, může být komentář:

```
/* toto je komentář */
```

Struktura programu: direktivy preprocesoru, deklarace, definice, funkce. V každém programu je právě jedna funkce hlavní (main), která se začne po spuštění programu vykonávat.

### 1.1.1 Deklarace

Deklarace jsou povinné. Deklaraci jednoduché proměnné tvoří specifikátor typu a jméno (identifikátor proměnné)

```
int a;                /* deklarace celočíselné proměnné a */  
int b=1;              /* definice proměnné b */
```

Podle umístění dělíme deklarace na globální (na začátku programu) a lokální (v těle funkce). Lokální proměnné nejsou implicitně inicializovány a obsahují náhodné hodnoty.

Specifikátory typu pro celá čísla: `int`, `char`, `short int` (nebo jen `short`), `long int` (nebo jen `long`).

Každý z nich může být `signed` (se znaménkem) nebo `unsigned` (bez znaménka), implicitně je `signed`.

Specifikátory typu pro racionální proměnné: `float` (32 bytů), `double` (64), `long double` (80).

U konstant je typ dán způsobem zápisu. Pomocí klíčového slova `const` můžeme deklarovat konstantní proměnnou, jejíž obsah nelze později měnit:

```
const float pi=3.14159;
```

### 1.1.2 Výrazy a přiřazení

Výrazy jsou v jazyce C tvořeny posloupností operandů a operátorů. Operátory dělíme podle arity (počet operandů) na unární, binární a ternární, podle funkce na aritmetické: `+`, `-`, `*`, `/`, `%` pro zbytek po dělení (operátor `/` má význam reálného nebo celočíselného dělení podle typů operandů), relační: `>`, `<`, `>=`, `<=`, `==` (rovnost), `!=` (nerovnost), logické: `||` (log.součet), `&&` (log.součin), `!` (negace). Jazyk C nezná logický typ, nenulová hodnota představuje `true`, nulová `false`.

#### Podmíněný operátor ? (jediný ternární operátor)

```
x=(a<b) ? a:b;
```

má stejný význam jako

```
if (a<b)      x=a; else x=b;
```

Obecně

```
v1 ? v2 : v3
```

`v1` je výraz, jehož hodnota je po vyhodnocení považována za logickou. Je-li `true`, vyhodnotí se výraz `v2` a vrátí se jeho hodnota, je-li `false`, pak se vyhodnotí `v3` a vrátí se jeho hodnota. `v2` a `v3` jsou jednoduché výrazy.

#### Operátory přiřazení

```
a=a+b;
```

```
a+=b;          /* má význam a=a+b; */
```

Na místě `+` může být `-`, `*`, `/`, `%`, `&` a další, o nichž zatím nebyla řeč.

#### Operátory inkrementace a dekrementace

```
a++;          /* postfixová verze */
```

```
--a;         /* prefixová verze */
```

Příklad:

```
a=10;
x=++a; /* x bude mít hodnotu 11, a taky */
y=a--; /* y=11, a=10 */
```

Unární operátory

Adresní operátor &, operátor dereference \*, unární +, unární -, logická negace ! a prefixová inkrementace ++ a dekrementace --. K postfixovým operátorům patří operátor přístupu k prvkům pole [ ], operátor volání funkce ( ), postfixová inkrementace ++ a dekrementace -- a operátory přístupu ke členům struktury, jimž se budu věnovat později.

**Operátor přetypování** ukáží na příkladu (i1 a i2 jsou celočíselné proměnné, ale chci reálné dělení):

```
f=(float) i1/i2;
```

**Operátor sizeof** pro zjištění velikosti: argumentem operátoru může být jak název typu, tak identifikátor proměnné.

1.1.3 Priorita a asociativita operátorů

| Priorita | Operátory                                             | Vyhodnocuje se |
|----------|-------------------------------------------------------|----------------|
| 1        | ( ) [ ] -> (funkce, index, přístup k prvku struktury) | zleva doprava  |
| 2        | ! - ++ -- + (typ)<br>* & sizeof (unární operátory)    | zprava doleva  |
| 3        | * / % (multiplikativní operátory)                     | zleva doprava  |
| 4        | + - (aditivní operátory)                              | zleva doprava  |
| 5        | << >> (operátory posunů)                              | zleva doprava  |
| 6        | < <= > >= (relační operátory)                         | zleva doprava  |
| 7        | == != (rovnost, nerovnost)                            | zleva doprava  |
| 8        | & (operátor bitového součinu)                         | zleva doprava  |
| 9        | ^ (exklusivní nebo)                                   | zleva doprava  |
| 10       | (operátor bitového součtu)                            | zleva doprava  |
| 11       | && (operátor logického součinu)                       | zleva doprava  |
| 12       | (operátor logického součtu)                           | zleva doprava  |
| 13       | ? : (ternární podmínkový operátor)                    | zprava doleva  |
| 14       | = += -= *= /= %=<br>>= &=  = ^= (oper. přiřazení)     | zprava doleva  |
| 15       | , (operátor čárky)                                    | zleva doprava  |

### 1.1.4 Příkazy a bloky

Napíšeme-li za výraz středník, stává se z něj příkaz, jako je tomu v následujících příkladech:

```
float x, y, z;  
x=0;  
a++;  
x=y=z;  
y=z=(f(x)+3); /* k hodnotě vrácené funkcí f je přičtena hodnota 3 */  
           /* součet je přiřazen jak proměnné z, tak y */
```

Příkazy v jazyce C můžeme sdružovat do tzv. bloků nebo složených příkazů. Blok může obsahovat deklarace proměnných na svém počátku a dále pak jednotlivé příkazy. Začátek a konec bloku je vymezen složenými závorkami.

Složené příkazy používáme tam, kde smí být použit pouze jeden příkaz, ale potřebujeme jich více. Za uzavírací složenou závorkou se nepíše středník.

#### Příkaz if

má dvě podoby:

```
if (výraz) příkaz  
if výraz příkaz1 else příkaz2;
```

Složitější rozhodovací postup můžeme realizovat konstrukcí if else if.

Každé else se váže vždy k nejbližšímu předchozímu if.

#### Příkaz switch a break

```
switch(výraz)  
{  
    case konst_výraz1:  
        /* příkazy, které se provedou, když výraz = výraz1 */  
        break;  
    case konst_výraz2:  
        /* příkazy, které se provedou, když výraz = výraz2 */  
        ...  
        break;  
    default:      /* příkazy, které se provedou, není-li výraz  
                  roven žádnému z předchozích konstantních výrazů */  
}
```

Příkaz break říká, že tok programu nemá pokračovat následujícím řádkem, nýbrž prvním příkazem za uzavírací složenou závorkou příkazu case.

V těle příkazu switch budou provedeny všechny vnořené příkazy počínaje tím, na který bylo předáno řízení, až do konce bloku (pokud některý z příkazů nezpůsobí něco jiného – např. break). Tím se switch značně liší od pascalského case.

### Příkaz while

`while (výraz) příkaz;`

Výraz za `while` představuje podmínku pro opakování příkazu. Není-li podmínka splněna už na začátku, nemusí se příkaz provést ani jednou. Je-li splněna, příkaz se provede a po jeho provedení se znovu testuje podmínka pro opakování cyklu.

### Příkaz do-while

Zajistí aspoň jedno provedení těla cyklu, protože podmínka opakování se testuje na konci cyklu.

`do příkaz while (výraz);`

### Příkaz for

`for (inicializační výraz; podmíněný výraz; opakovaný výraz) příkaz`

je ekvivalentní zápisu:

```
inicializační výraz;
while (podmíněný výraz)
{
    příkaz
    opakovaný výraz
}
```

Inicializační výraz může být vypuštěn, zůstane po něm však středník. Stejně může být vynechán i podmíněný výraz a opakovaný výraz. Příkaz `continue` je možno použít ve spolupráci se všemi uvedenými typy cyklů. Ukončí právě prováděný průchod cyklem a pokračuje novým průchodem. Podobně i příkaz `break` může být použit ve všech typech cyklů k jejich ukončení.

### Příkaz goto a návěští

Příkaz `goto` přenesení běh programu na místo označené návěstím (identifikátor ukončený dvojtečkou). Jsou situace, kdy může být užitečný, např. chceme-li vyskočit z vnitřního cyklu z více vnořených cyklů.

### Prázdný příkaz

`;`

Lze ho použít všude tam, kde je prázdné tělo.

## 1.1.5 Preprocesor

Preprocesor zpracuje zdrojový text programu před překladačem, vypustí komentáře, provede záměnu textů, např. identifikátorů konstant za odpovídající číselné hodnoty a vloží texty ze specifikovaných souborů. Příkazy pro preprocesor začínají znakem `#` a nejsou ukončeny středníkem. Nejdůležitějšími příkazy jsou `#define` a `#include`.