

Programování v jazyce C pro chemiky (C2160)

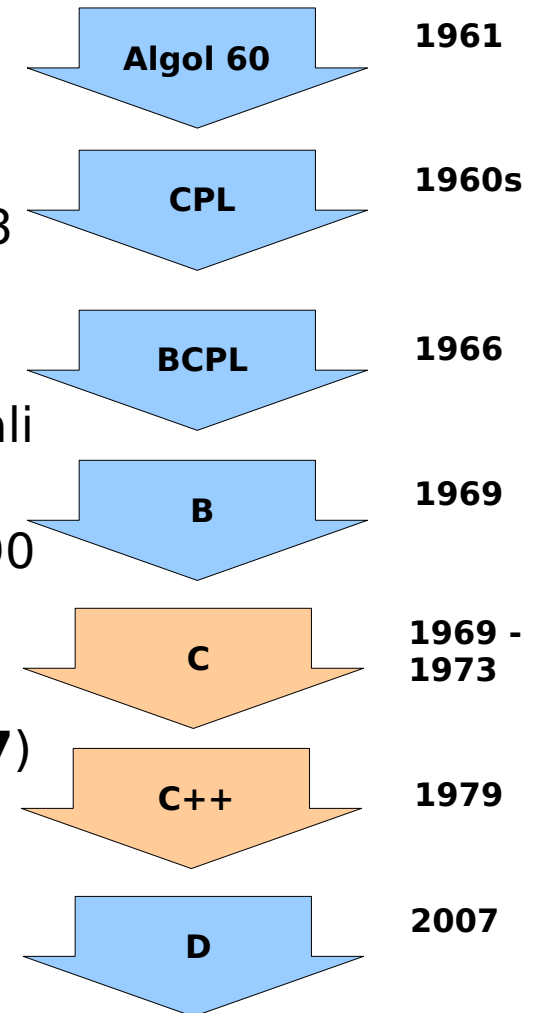
1. Úvod do C, příkaz podmínky

Historie jazyka C

- Jazyk C vyvinul Dennis Ritchie v AT&T Bell Labs v letech 1969–1973
- Počátek vývoje C úzce spjat s počátky vývoje Unixu – zpočátku byl Unix napsaný v assembleru, ale v roce 1973 byl přepsán do C
- 1978 – Brian Kernighan (**AWK**) a Dennis Ritchie publikovali knihu "The C Programming Language" (**K&R C**)
- 1989–90 – standard ANSI X3.159-1989, ISO/IEC 9899:1990 – **C89, C90, ISO C, ANSI C**
- 1999 – ISO/IEC 9899:1999 – **C99**
- 2011 – ISO/IEC 9899:2011 – **C11**, 2017 oprava chyb (**C17**)

Zkratky:

ANSI – American National Standards Institute
ISO – International Organization for Standardization
IEC – International Electrotechnical Commission



Jazyk C

- Jazyk C je široce používaný pro tvorbu operačních systémů, aplikačních programů, počítačových her, programování elektronických zařízení
- Je široce využíván ve vědeckých oborech pro tvorbu výpočetně náročných programů (společně s jazykem Fortran)
- Ze syntaxe jazyka C vycházejí C++, Java, C#, JavaScript, PHP, C shell (csh, tcsh)

Výhody:

- Vysoká výpočetní rychlost a paměťová efektivita programů
- Možnost nízkoúrovňových operací pro přímou komunikaci s hardwarem (operační systémy, ovladače hardwaru)
- Široká dostupnost kvalitních překladačů a knihoven

Nevýhody:

- Relativně složitý jazyk ⇒ nepříliš vhodný pro začátečníky (jako první programovací jazyk)
- Nutnost kompilace zdrojového kódu po každé změně v programu

Kompilace programu v C

- Jazyk C patří mezi takzvané **kompilované jazyky**, to znamená, že po vytvoření souboru se zdrojovým kódem programu v C musíme spustit **překladač** (kompilátor), který vygeneruje soubor se spustitelným kódem
- Pod Linuxem je standardně dostupný kompilátor **gcc**, kompilaci spustíme následujícím způsobem:

```
gcc -o spustitelny_soubor zdrojovy_soubor.c
```

Zdrojový soubor

obsahující kód programu v jazyce C vytvoříme v textovém editoru (soubory mají obvykle koncovku **.c**)

Spustíme kompilátor

který ze zdrojového souboru *.c vygeneruje spustitelný soubor

Spustitelný soubor

obsahuje instrukce pro procesor a lze ho spustit např. z příkazového řádku

Programátorský editor Kate

- Editor *Kate* je programátorský editor který umožňuje automatické odsazování textu, barevné označení klíčových slov jazyka, vyznačení párování závorek a pod.
- Pro správné odsazování textu odpovídající jazyku C je potřeba aby editovaný soubor měl koncovku **.c**
- Kontrola párování závorek: pokud umístíme textový kurzor na složenou nebo obyčejnou závorku, označí se příslušná párová závorka (tučně a případně žlutou barvou)
- Multiplatformní, dostupný ve Windows Store

```
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

Kompilace programu v C - příklad

```
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

- Vytvořte složku do které budete ukládat programy ze cvičení
- Spusťte editor *kate*, napište výše uvedený program v jazyku C a uložte ho do souboru *helloworld.c*
- V okně terminálu přejděte do složky s vytvořeným souborem *helloworld.c* a spusťte kompilátor: *gcc -o helloworld helloworld.c*
- Kompilátor vygeneruje spustitelný soubor *helloworld*
- V okně terminálu spusťte program: *./helloworld*

Chybové hlášky kompilátoru

```
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

- Za funkci `printf` přidáme závorku navíc
- Při překladu nahlásí kompilátor chybu
- Uveden je popis chyby a číslo řádku
- POZOR! – číslo řádku nemusí odpovídat skutečné pozici chyby ale odpovídá místu od kterého překlad nedává kompilátoru smysl. Skutečná chyba se může nacházet na některém z předešlých řádků.
- Příklad: vynechejte středník za funkcí `printf` – chyba bude nahlášena na následujícím řádku
- Čísla řádků lze v *Kate* zobrazit: [Menu: View / Show Line Numbers](#)

Struktura programu v C

Program pro sčítání dvou celých čísel:

Funkce `main()` vrací celočíselnou (**integer**) hodnotu

Na začátku programu jsou vloženy tzv. hlavičkové soubory. Soubor `stdio.h` nám umožní používat funkci `printf()` pro výstup na obrazovku

Funkcí `main()` začíná každý program v C. Kód funkce je uzavřený ve složených závkách `{}`

Funkce `printf()` provede výpis textu a obsahu proměnných

```
#include <stdio.h>
```

Definujeme 3 proměnné celočíselného (**integer**) typu

```
int main()
```

```
{
```

```
    int a, b, c;
```

```
    a = 2;
```

```
    b = 3;
```

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

Proměnné **a** přiřadíme hodnotu **2**, proměnné **b** **3**

Proměnné **c** přiřadíme součet hodnot proměnných **a** a **b**

Opuštění funkce `main` a ukončení programu

```
    printf("Soucet %i a %i je %i\n", a, b, c);
```

```
    return 0;
```

```
}
```

Text vypisovaný na obrazovku. Formátovací prvky `%i` slouží k výpisu obsahu proměnných. Speciální znak `\n` slouží k odřádkování (tj. přechodu na nový řádek).

Obsah proměnných **a**, **b**, **c** je předán funkci `printf()`, která zajistí jejich výpis. Každé předávané proměnné musí odpovídat jeden formátovací prvek `%i`.

Struktura programu v jazyce C

- Veškeré příkazy jazyka C jsou zapsány ve funkcích
- Program začíná vždy funkcí `main()`, tuto funkci tedy musí nezbytně obsahovat všechny programy napsané v jazyce C
- Kód funkce je uzavřen ve složených závorkách `{}`
- Každý příkaz je zakončen středníkem
- Jazyk C je `case-sensitive`, tj. rozlišuje velká a malá písmena

Komentáře v C

- 1. typ: použití dvou lomítek, vše za nimi až do konce řádku je komentář (**C99**): *// Vše za dvěma lomítky je komentář*
- 2. typ: lze použít přes více řádků: */* Nějaký text komentáře */*

```
// Komentar muze byt i mimo funkci

int main()
{
    /* Samozrejme muze byt i ve funkci. Tento typ komentare
       s lomítkem a hvězdičkou muze pokračovat i přes více radku */
    int a, /* Dokonce i tady muze byt */ b, c;
    a = 2; // Vse za dvema lomítky az do konce radku je komentar
    b = 3; //Neni treba delat za lomítka mezeru, ale je toprehlednejsi
    c = a + b;
    //c = a - b; komentare lze pouzít i k docasnemu zneplatneni kodu

    printf("Soucet %i a %i je %i\n", a, b, c);
    /* Pokud je komentar viceradkovy, dava se
       * nekdy na zacatek vsech radku hvězdička.
       * Neni to sice nutne, ale znacne to zprehlednuje
       * text komentare, hlavne pokud je hodne dlouhy.
       */

    return 0;
}
```

Mezery v C

- Většinu mezer lze vynechat nebo libovolně přidávat, hlavně před a po znacích `=+-*/[](){}.,;:?'<>&|~%"`
- Mezery nelze vkládat doprostřed názvů (např. funkcí, proměnných, typů a pod.)
- Mezera je nezbytná k oddělení jména typu od jména proměnné nebo funkce a také za příkazem `return`
- Z důvodů přehlednosti je vhodné mezery hojně používat

```
int _main_ ( _  
{  
    int _a, _b, _count_ ;  
    a _= _2_ ;  
    b _= _3 ;  
    c _= _a_ + _b ;  
  
    printf_ ( _"Soucet %i a %i je %i\n", _a, _b, _c_ ) _ ;  
  
    return _0 ;  
}
```

 mezery povinné
 mezery nepovinné

Proměnné

- Každá proměnná musí být před prvním použitím definována tj. uveden typ a její jméno
- Definici proměnné je vhodné uvádět na začátku funkce
- Základní typy proměnných:
 - int** - celá čísla (**integer**)
 - float** - desetinná čísla (**floating point number**)
 - char** - znak (**character**)
- Při definici uvedeme jméno typu, mezeru a seznam proměnných oddělených čárkou
- Počáteční hodnotu proměnné lze nastavit pomocí = (mluvíme o tzv. inicializaci proměnné)

```
int a, b, c; // definujeme tři celociselné proměnné a, b, c
int d = 5; // definujeme proměnnou d a inicializujeme ji hodnotou 5
float m, p; // definujeme necelociselné proměnné m a p
float g = 1.75; // definujeme proměnnou g a inicializujeme ji hodnotou 1.75
char z; // definujeme znakovou proměnnou z

int teplota, vlhkost; // jméno proměnné může být i delší než jeden znak
int partial_charge3; // můžeme používat písmena, číslice, podtržítka, ne
// však mezeru
int MaxValue; // lze použít i velká písmena
int maxValue; // toto je již jiná proměnná (jazyk C je case-sensitive!)
```

Konstanty

- Konstanty jsou číselné nebo znakové hodnoty, které používáme např. pro přiřazení hodnot proměnným
- Nejpoužívanější jsou následující konstanty:
 - **Celočíselné konstanty** zapisujeme jako běžná čísla (např. 1, -4, 178)
 - **Reálné konstanty** zapisujeme s desetinnou tečkou (1.14, -58.146)
 - **Znakové konstanty** zapisujeme jako znak uvedený **v apostrofech** (např. 'A', 'r')
- Konstanty používáme např. pro inicializaci proměnných; pro každou proměnnou používáme odpovídající typ konstanty (tj. pro reálnou proměnnou použijeme reálnou konstantu, pro znakovou proměnnou použijeme znakovou konstantu atd.)
- Všechny proměnné bychom měli vždy inicializovat vhodnou implicitní hodnotou (často 0)

```
int d = 5;    // inicializace celociselné proměnné celociselnou konstantou
int a = 0, b = 0;
float g = 1.75; // inicializace reálné proměnné reálnou konstantou
float m = 0.0, n = 0.0;
char z = 'D'; // inicializace znakové proměnné znakovou konstantou
```

Přiřazovací příkaz a aritmetické operátory

- Přiřazovací příkaz =
- Aritmetické operátory + - * /
- Operátor % poskytuje při dělení celých čísel zbytek po dělení (tzv. modulo)
- Pro určení priority operací lze používat závorky

```
int a, b, c, d; // definice promenných a, b, c
a = 5;         // promenne a priradi hodnotu 5
b = 7;         // promenne b priradi hodnotu 7
c = 13;        // promenne c priradi hodnotu 13

d = a + b;     // promenne d priradi hodnota souctu a plus b
d = a - b;
d = a * b;
d = a / b;     // POZOR!: deleni nulou zpusobi okamzite ukonceni programu
d = c % a;     // Zbytek po deleni cisla 13 cislem 5 coz je 3

d = a + b / c; // Provede se deleni b/c, pak se pricte a
d = (a + b) / c; // Secte se a plus b a pak se to vydeli c
```

Funkce printf() - výstup textu na obrazovku

- Funkce printf() slouží k výpisu textu na obrazovku (terminál)
- Prvním parametrem funkce je text který bude vypsán na obrazovku (zapisuje se v uvozovkách)
- Dále může následovat seznam libovolného počtu proměnných jejichž hodnoty mají být vypsány
- Místo v textu, kde se vypíše hodnota proměnných se specifikuje pomocí formátovacích prvků:
 - **%i** pro proměnné typu int (častější ekvivalent je **%d**)
 - **%f** pro proměnné typu float
 - **%c** pro proměnné typu char
- Speciální znak **\n** způsobí přechod na nový řádek

```
int a = 3, b = 6, c = 9;
float m = 2.71, n = 14.3;
char c1 = 'D', c2 = 'j';

printf("Jednoduchy text\n");
printf("Cislo a ma hodnotu %i zatimco b je %i\n", a, b);
printf("Vypis celeho cisla %i a desetinneho cisla %f\n", c, m);
printf("Vypis dvou znaku za sebou: %c%c\n", c1, c2);
```

Použití matematických funkcí

- Pro zpřístupnění matematických funkcí přidáme na začátek programu: `#include <math.h>`
- Při kompilaci je potřeba přidat matematickou knihovnu pomocí **lm** (l jako **library** a m jako **math**), např:
`gcc -o mujprogram mujprogram.c -lm`
- Nejpoužívanější matematické funkce:
 - sqrt(x)** – odmocnina z x
 - pow(x, y)** – x umocněno na y (u druhé, třetí mocniny však dáváme přednost jednoduchému násobení)
 - exp(x)** – e na x
 - log(x)** – přirozený logaritmus x
 - cos(x)** – kosinus x
 - sin(x)** – sinus x

```
int n = 3;
float a = 7.3, b = 2.5, result = 0.0;

result = sqrt(a);
result = pow(a, n) + 7.4;
result = a + cos(b);
result = (sqrt(a) + sqrt(b)) / 2;
```


Funkce scanf () - čtení vstupu z klávesnice

- Funkce scanf () slouží k získání vstupu z klávesnice
- Prvním parametrem funkce je formátovací text, který obsahuje jeden nebo více formátovacích prvků
- Do formátovacího textu **nezadáváme žádný text**, používáme jen formátovací prvky
- Pokud použijeme více než jeden formátovací prvek, oddělujeme je zpravidla mezerou – mezeru pak musíme použít pro oddělení hodnot zadávaných z klávesnice
- Před načítáním hodnot pomocí scanf() je zpravidla vhodné vypsát na obrazovku výzvu k zadání příslušných hodnot, proto nejdříve zavoláme printf(), která text vypíše, a pak načítáme pomocí scanf()
- Za formátovací text funkce scanf () zapíšeme tolik proměnných, kolik bylo použito formátovacích prvků
- **POZOR:** Před název proměnné musíme vložit znak **&**

```
int a = 0;
float m = 0.0, n = 0.0;

printf("Zadej cele cislo a stiskni Enter: ");
scanf("%i", &a);

printf("Zadej dve desetinna cisla oddelena mezerou: ");
scanf("%f %f", &m, &n);
```

Příklad použití funkce scanf()

Program demonstrující čtení vstupu z klávesnice:

- Program si vyžádá dvě desetinná čísla A a B a vypíše jejich součet

```
int main()
{
    float a = 0.0, b = 0.0, c = 0.0;

    printf("Zadej A a B oddelene mezerou a stiskni Enter: ");
    scanf("%f %f", &a, &b);

    c = a + b;

    printf("Soucet %f a %f je %f\n", a, b, c);

    return 0;
}
```

Příkaz podmínky **if**

- Příkaz slouží k podmíněnému provádění příkazů
- Zapisuje se:
if (*podmínka*) *příkaz*;
- Můžeme specifikovat i příkaz který se provede při nesplnění podmínky:
if (*podmínka*) *příkaz1*; **else** *příkaz2*;

```
int a = 4, b = 0;

if (a > 3)
    b = 10;    // Tento prikaz se provede pouze pokud a > 3

if (a > 8)
    b = 10;    // Tento prikaz se provede pouze pokud a > 8
else
    b = 0;     // Tento prikaz se provede pouze pokud neni a > 8
```

Příkaz podmínky `if`

- Pokud potřebujeme podmíněně vykonat více než jeden příkaz, použijeme složené závorky

```
int a = 4, b = 0, c = 0, d = 0;

if (a > 3)
{
    b = 10;
    c = a + b;
}

if (a > 8)
{
    b = 5;
    c = a - b;
}
else
{
    b = 7;
    d = a + b;
}
```

Příkaz podmínky `if`

- Složené závorky je používat i pro jeden příkaz, hlavně je-li přítomen `else` blok nebo jde o vnořenou podmínku
- Bez složených závorek může přidání nového příkazu do těla `if/else` úplně změnit význam podmínek

```
int a = 4, b = 1;

if (a > 3)
    if (a > 8)
        b = 3;
else
    b = 2;
/* Jakou hodnotu ma b v tomto miste? */

if (a > 3)
{
    if (a > 8)
    {
        b = 3;
    }
else
    {
        b = 2;
    }
}
```

Příkaz podmínky `if`

- Závorky a „else“ můžeme umístit dle vkusu, hlavně ale konzistentně!

```
if (a > 3)
{
    if (a > 8)
    {
        b = 3;
    }
    else
    {
        b = 2;
    }
}

if (a > 3) {
    if (a > 8) {
        b = 3;
    } else {
        b = 2;
    }
}
```

Příkaz podmínky `if` - konstrukce podmínky

- Pro konstrukci podmínky používáme následující operátory:

`==` rovnost
`!=` nerovnost
`<` menší
`<=` menší nebo rovno
`>` větší
`>=` větší nebo rovno
`||` nebo (logické OR)
`&&` a zároveň (logické AND)

- Operátory se vyhodnocují podle priority

- Zjednodušená tabulka priorit operátorů:

`<` `<=` `>` `>=` nejvyšší priorita

`==` `!=`

`&&`

`||`

nejnižší priorita

- Prioritu lze ovlivnit pomocí závorek

- Doporučení: prioritu raději přesně specifikovat pomocí závorek (menší pravděpodobnost, že uděláme chybu)

Příkaz podmínky `if` - konstrukce podmínky

Následující ukázka je zjednodušená, příkazy za podmínkami nejsou uvedeny

```
int a = 4, b = 5, c = 9;
char ch1 = 'A', ch2 = 'D';

if (a == 3)           // Jednoduchá podmínka
if (a != 3)           // Jednoduchá podmínka
if (b <= 3)           // Jednoduchá podmínka
if (ch1 == 'B')       // Jednoduchá podmínka
if (ch1 != 'h')       // Jednoduchá podmínka

if (a > 3 && b <= c)   // Trochu složitější podmínka
if ((a > 3) && (b <= c)) // Tato podmínka, ale s použitím závorek

if (a > 3 || b != c && b <= c) // Prioritu má && před ||
if (a > 3 || (b != c && b <= c)) // Totéž s použitím závorek
if ((a > 3 || b != c) && b <= c) // Tady závorky změnilý pořadí vyhodnocení
```


Příklad použití podmínky

Program načte z klávesnice celá čísla a, b a vypíše, které je větší

```
int main()
{
    int a = 0, b = 0;

    printf("Zadej A a B oddelene mezerou a stiskni Enter: ");
    scanf("%i %i", &a, &b);

    if (a == b)
    {
        printf("Vysledek: A je rovno B\n");
    }
    else
    {
        printf("Vysledek: A je ruzne od B\n");
    }

    return 0;
}
```

Dodržujte následující pravidla

- Všechny **proměnné vždy inicializujte** vhodnou implicitní hodnotou při její definici (nejčastěji 0)
- Dbejte na **správné odsazování textu**. K odsazování by mělo docházet v editoru *Kate* automaticky. Není-li text správně zarovnan, označte veškerý text a pak aplikujte Menu: Tools/ Align. Pokud nedojde ke správnému zarovnání, příčinou může být špatná koncovka souboru (musí být *.c*), nesprávně spárované závorky nebo chybějící středník na konci příkazu.
- **Úlohy odevzdávejte do odevzdáárny** předmětu C2160. V odevzdáárně naleznete složky pro jednotlivá cvičení. Do příslušné z nich vkládejte soubory jednotlivých úloh pojmenované tak, aby z názvu bylo možné určit číslo úlohy. Není třeba do názvu souboru vkládat vaše jméno, to doplní IS sám. **Odevzdávejte pouze soubory s kódem v C**, nikoliv spustitelné soubory.

Úlohy

1. Vytvořte program pro řešení kořenů kvadratické rovnice. Program si od uživatele vyžádá zadání koeficientů a , b , c , spočítá kořeny a vypíše je na obrazovku. Ošetřete situaci, kdy je diskriminant záporný nebo koeficient a je roven 0. **1 bod**
2. Vytvořte program pro sčítání čísel. Čísla budou zadávána z klávesnice. Program se na začátku zeptá, kolik čísel budete chtít sčítat (max. pět). Potom si vyžádá zadání čísel z klávesnice. Nakonec zobrazí výsledek. **1 bod**
3. Vytvořte program který vypíše seznam možných operací (sčítání, odčítání, násobení, dělení) označených čísly 1 až 4. Uživatel si vybere, kterou operaci chce provést. Potom bude vyzván k zadání dvou čísel. Nakonec se vypíše výsledek. **1 bod**
4. Modifikujte předchozí program tak, že volba operace se neprovede pomocí čísla 1 až 4 ale zadáním jednoho ze znaků +, -, *, /. **nepovinná, 1 bod**