# Sbírka úloh z jazyka C

Petr Krajča, Tomáš Kühr, Vilém Vychodil

UNIVERZITA PALACKÉHO V OLOMOUCI

# Obsah

# 1 Základy jazyka C

## 1.1 Hello World

**Procvičované učivo:** seznámení se s vývojovým prostředím, základní výstup na obrazovku

Napište v jazyku C program, který vypíše na obrazovku text "*Hello, world!*".

**Příklad použití:**

`./hello` (OS Linux)

`hello.exe` (OS Windows)

**Příklad výstupu:**

```
Hello, world!
```

**Povolené knihovny:** stdio.h, stdlib.h

## 1.2 Práce s proměnnými

**Procvičované učivo:** proměnné, konstanty, výpis na obrazovku

Vytvořte v jazyku C program, v němž budou definovány a inicializovány proměnné různých datových typů. Hodnoty proměnných pak vypište na obrazovku.

**Příklad výstupu:**

```
Hodnota promenne cislo je: 3
Hodnota promenne des_cislo je: 3.45
Hodnota promenne muj_znak je: +
Hodnota promenne male_cislo je: 1.2e-10
```

**Povolené knihovny:** stdio.h, stdlib.h

## 1.3 Osmičkový a šestnáctkový výstup

**Procvičované učivo:** základní vstup a výstup

Napište v jazyku C program, který načte číslo v desítkové soustavě a vypíše ho na obrazovku v osmičkové a šestnáctkové soustavě.

**Příklad výstupu:**

```
Zadejte cislo: 12
Cislo 12 odpovida cislu 14 v osmickove soustave
a cislu C v sestnactkove soustave.
```

**Povolené knihovny:** stdio.h, stdlib.h

# 2 Operátory

## 2.1 Obsah obdélníku

**Procvičované učivo:** operátor přiřazení, aritmetické operátory, základní vstup a výstup

Napište v jazyku C program, který výpočítá obsah obdélníka a vypíše jej na obrazovku. Vstupem programu jsou velikosti stran obdélníku.

**Příklad výstupu:**

```
Zadejte stranu a: 8
Zadejte stranu b: 5
Obsah obdelniku je: 40
```

**Povolené knihovny:** stdio.h, stdlib.h

## 2.2 Sestrojitelnost trojúhelníku

**Procvičované učivo:** podmínkový operátor, logické operátory, operátory porovnání, základní vstup a výstup

Napište v jazyku C program, který ověří, jestli lze sestrojit trojúhleník se zadanými velikostmi stran. Vstupem programu jsou velikosti stran trojúhelníku.

**Příklad výstupu:**

```
Zadejte stranu a: 8
Zadejte stranu b: 5
Zadejte stranu c: 20
Trojuhelnik nelze sestrojit.
```

**Povolené knihovny:** stdio.h, stdlib.h

## 2.3 Celá část

**Procvičované učivo:** operátor přiřazení, přetypování, základní vstup a výstup

Napište v jazyku C program, který načte desetinné číslo, vypočítá celou část tohoto čísla a vypíše ji na obrazovku.

**Příklad výstupu:**

```
Zadejte cislo: 8.024
Cela cast cisla 8.024 je 8.
```

**Povolené knihovny:** stdio.h, stdlib.h

## 2.4 Zaokrouhlení

**Procvičované učivo:** aritmetické operátory, operátor přiřazení, přetypování, podmínkový operátor, základní vstup a výstup

Napište v jazyku C program, který načte desetinné číslo a požadovanou "přesnost" a vypíše na obrazovku toto číslo zaokrouhlené s danou přesností.

**Příklad výstupu:**

```
Zadejte cislo: 8.027
Zadejte presnost: 0.01
Cislo po zaokrouhleni je: 8.03
```

**Povolené knihovny:** stdio.h, stdlib.h

# 3 Větvení programu

## 3.1 Rozpoznání znaku

**Procvičované učivo:** větvení if, větvení switch , základní vstup a výstup

Napište v jazyku C program, který pro zadaný znak **slovy** vypíše, o jaký znak se jedná...

Pro malá písmena vypíše program text *"male pismeno"* a daný znak, pro velká písmena vypíše text *"velke pismeno"* a daný znak, pro číslice vypíše *"cislice"* a daný znak. Dále pro znaky *"!"*, *"?"*, *"*"*, *"@"*, *"#"*, *"^ "* vypíše odpovídající text: *"vykricnik"*, *"otaznik"*, *"hvezdicka"*, *"zavinac"*, *"krizek"*, *"striska"*. Pokud se jedná o jiný než výše uvedený znak, vypíše program text *"jiny znak"*.

**Příklad výstupu:**

```
Zadejte znak: c
Zadany znak je: male pismeno c
```

**Povolené knihovny:** stdio.h, stdlib.h

## 3.2 Maximum

**Procvičované učivo:** větvení if, základní vstup a výstup

Napište v jazyku C program, který po zadání trojice čísel určí největší z nich a vypíše jej na obrazovku.

**Příklad výstupu:**

```
Zadejte prvni cislo: 4
Zadejte druhe cislo: 8
Zadejte treti cislo: 1
Nejvetsi cislo je: 8
```

**Povolené knihovny:** stdio.h, stdlib.h

## 3.3 Výpočet progresivní daně

**Procvičované učivo:** větvení if, přiřazení, aritmetické operátory, základní vstup a výstup

Napište v jazyku C program, který po zadání mzdy vypočítá a na obrazovku vypíše výši odpovídající daně.

Pro účel této úlohy uvažujme progresivní zdanění ve výši 10 % pro příjem do 10000, 20 % pro příjem od 10000 do 20000 a 30 % pro příjem nad 20000. Například, pokud máme hrubou mzdu 24000, bude se prvních 10000 danit 10 % (tj. daň z této části mzdy je 1000), dalších 10000 se daní 20 % (daň z této části je 2000) a zbývající 4000 se daní 30 % (daň je 1200). Celkovou výši daně pak vypočítáme jako součet jednotlivých "částečných" daní (tj. celková daň 4200).

**Příklad výstupu:**

```
Zadejte mzdu: 12000
Odpovidajici dan je: 1400
Zadejte mzdu: 33353
Odpovidajici dan je: 7005.9
```

**Povolené knihovny:** stdio.h, stdlib.h

# 4 Cykly

## 4.1 Násobilka

**Procvičované učivo:** cykly, aritmetické operátory, operátor přiřazení, operátory porovnání, základní vstup a výstup

Napište v jazyku C program, který pro zadané číslo vypíše na obrazovku všechny jeho násobky menší nebo rovny číslu 100.

**Příklad výstupu:**

```
Zadejte cislo: 7
Nasobky zadaneho cisla:
 7, 14, 21, 28, 35, 42, 49, 56, 63, 70,
77, 84, 91, 98
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** výpis prvních 100 násobků zadaného čísla, možnost specifikovat mezní hodnotu uživatelem

## 4.2 Prvočísla

**Procvičované učivo:** cykly, aritmetické operátory, operátor přiřazení, operátory porovnání, základní vstup a výstup

Napište v jazyku C program, který vypíše na obrazovku všechna prvočísla menší než 100.

**Příklad výstupu:**

```
Prvocisla:
 2,  3,  5,  7, 11, 13, 17, 19, 23, 29,
31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** možnost specifikovat mezní hodnotu uživatelem

## 4.3 Čtverec

**Procvičované učivo:** cykly, aritmetické operátory, operátor přiřazení, operátory porovnání, základní vstup a výstup

Napište v jazyku C program, který vykreslí pomocí znaku "*" na obrazovku čtverec zadané velikosti.

**Příklad výstupu:**

```
Zadejte velikost ctverce: 5

*****
*   *
*   *
*   *
*****
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** vykreslování obdélníku, pravoúhlého trojúhelníku, ...

# 5 Jednorozměrná pole

## 5.1 Obrácení pole

**Procvičované učivo:** jednorozměrná pole, cykly, operátory porovnání, přiřazení, základní výstup

V programu deklarujte jednorozměrné pole, naplněte ho čísly a vypište hodnoty čísel v tomto poli na obrazovku. Poté obraťte pořadí čísel v tomto poli (tj. první bude původně poslední číslo, druhé původně předposlední atd.) a hodnoty ve změněném poli vypište na obrazovku.

Není dovoleno použití druhého pole, ani znovunaplnění pole hodnotami v opačném pořadí. Část zdrojového kódu, která zaměňuje pořadí prvků v poli, musí být zcela nezávislá na tom, jakými hodnotami bylo pole původně naplněno.

**Příklad výstupu:**

```
Puvodni hodnoty: 1, 2, 3, 4, 5, 6, 7
Nove hodnoty:    7, 6, 5, 4, 3, 2, 1
```

**Povolené knihovny:** stdio.h, stdlib.h

## 5.2 Eratosthenovo síto

**Procvičované učivo:** jednorozměrná pole, cykly, aritmetické operátory, operátory porovnávání, přiřazení, základní výstup

V jazyku C vytvořte program, který pomocí algoritmu Eratosthenova síta určí a vypíše všechna prvočísla menší než číslo 100.

**Algoritmus Eratosthenova síta:**

1. Vytvoříme pole všech čísel od 2 do požadovaného maximálního zkoumaného čísla.

2. Procházíme pole od začátku, dokud nenajdeme nevyškrtnuté číslo. Toto číslo je prvočíslem, můžeme jej proto vypsat na obrazovku.

3. Vyškrtáme z pole všechny násobky právě nalezeného prvočísla (např. změnou hodnoty na 0).

4. Pokračujeme krokem 2, dokud zbývají nějaká nevyškrtnutá čísla.

**Příklad výstupu:**

```
Prvocisla:
 2,  3,  5,  7, 11, 13, 17, 19, 23, 29,
31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97
```

**Povolené knihovny:** stdio.h, stdlib.h

## 5.3 Aritmetický průměr pole

**Procvičované učivo:** jednorozměrná pole, cykly, aritmetické operátory, operátor přiřazení, základní výstup

V programu deklarujte jednorozměrné pole a naplněte ho čísly. Poté vypočítejte aritmetický průměr čísel v tomto poli a vypište jej na obrazovku.

**Příklad výstupu:**

```
Pole obsahuje cisla: 1, 2, 3, 4, 5, 6, 7
Prumer pole je: 4
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** výpočet sumy, maxima, minima

# 6 Funkce

## 6.1 Suma pole

**Procvičované učivo:** funkce, cykly, aritmetické operátory, operátor přiřazení, základní výstup

Definujte v jazyku C funkci odpovídající deklaraci `double suma_pole(double pole[], int pocet)`, která vypočítá a vrátí součet čísel v daném poli. Parametr `pole` slouží pro předání pole čísel, parametr `pocet` pak odpovídá počtu čísel v předávaném poli.

**Příklad výstupu:**

```
Pole obsahuje cisla: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Suma pole je: 55
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** výpočet průměru, maxima, minima

## 6.2 Převody čísel do desitkové soustavy

**Procvičované učivo:** funkce, práce s textovými řetězci, cykly, větvení, konstrukce složených podmínek

Napište v jazyku C funkci odpovídající deklaraci `int do_desitkove(char cislo[], int zaklad)`, která dané číslo převede do desítkové soustavy. Převáděné číslo je zadáno jako textový řetězec tvořený jednotlivými číslicemi (parametr `cislo`). Parametr `zaklad` pak udává hodnotu základu soustavy, ze které převádíme. Pro číslice soustav se základem větším než deset používáme znaky velkých písmen (např. "*A*" pro číslici s hodnotou 10).

**Příklad výstupu:**

```
Cislo A1B v soustave o zakladu 12
odpovida cislu 1463 v desitkove soustave.
```

**Povolené knihovny:** stdio.h, stdlib.h, string.h

# 7 Strukturované datové typy

## 7.1 Studenti

**Procvičované učivo:** strukturované datové typy, funkce, větvení

Vytvořte v jazyku C strukturovaný datový typ `datum` se členy `den`, `mesic` a `rok`. Poté vytvořte strukturovaný typ `student` se členy `jmeno`, `prijmeni` a `narozen`. Pro reprezentaci jednotlivých členů struktur zvolte vhodné datové typy.

Dále napište funkci `int porovnej_vek(student s1, student s2)`, která porovná věk (resp. datum narození) daných studentů a vrátí hodnotu -1 v případě, že první student je starší, 1 v případě, že druhý student je starší a 0 v případě shodného data narození u obou studentů. Podle návratové hodnoty funce `porovnej_vek` pak ve funkci `main` vypište vhodný text na obrazovku.

**Příklad výstupu:**

```
Pepa Stary je starsi nez Adam Novak.
```

**Povolené knihovny:** stdio.h, stdlib.h

## 7.2 Součet zlomků

**Procvičované učivo:** strukturované datové typy, funkce

Vytvořte v jazyku C strukturovaný datový typ `zlomek` se členy `citatel` a `jmenovatel`. Dále napište funkci `zlomek soucet(zlomek a, zlomek b)`, která vypočítá a vrátí součet předaných zlomků v základním tvaru. Vykrácení výsledného zlomku do základního tvaru docílíte vydělením čitatele i jmenovatele jejich největším společným dělitelem, který můžete určit například použitím **Euklidova algoritmu**.

**Příklad výstupu:**

```
Soucet zlomku 2/3 a -1/6 je: 1/2
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** výpočet rozdílu, součinu nebo podílu dvojice zlomků

# 8 Ukazatele

## 8.1 Porovnání textových řetězců

**Procvičované učivo:** ukazatele, práce s textovými řetězci, funkce, cykly

Napište v jazyku C funkci `int porovnej(char *t1, char *t2)`, která porovná předané textové řetězce a vrátí -1, pokud je první řetězec menší než druhý, 0, pokud jsou řetězce shodné, nebo 1, pokud je druhý řetězec menší než první. Při práci s textovými řetězci používejte výhradně ukazatele, operátor dereference a pointerovou aritmetiku.

Porovnávání řetězců by mělo být lexikografické, tj. obdobné uspořádání slov ve slovníku. Budou tedy porovnávány jednotlivé odpovídající si dvojice znaků (*i*-tý znak prvního řetězce s *i*-tým znakem druhého řetězce) počínaje prvními znaky obou řetězců, první rozdílná dvojice znaků pak určí výsledek porovnání obou řetězců. Tento způsob porovnání textových řetězců plně odpovídá funkci `strcmp`.

**Příklad výstupu:**

```
Slovo "ahoj" je větší než slovo "abcde".
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** vytvoření jiných funkcí ze string.h: strchr, strrchr...

## 8.2 Hledání podřetězce zprava

**Procvičované učivo:** ukazatel, práce s textem, funkce, cykly

Napište v jazyku C funkci `char *strrstr(const char *text, const char *hledany)`, která v daném textovém řetězci `text` vyhledá první výskyt zadaného podřetězce `hledany` zprava. Funkce vrací ukazatel na první znak nalezeného podřetězce nebo konstantu `NULL`, pokud podřetězec `hledany` nebyl nalezen. Vytvořenou funkci otestujte ve funkci `main`.

**Příklad výstupu:**

```
Text: "Ahoj svete!"
Hledame: "svet"
Vraceny ukazatel: "svete!"
```

**Povolené knihovny:** stdio.h, stdlib.h

# 9 Dynamická práce s pamětí

## 9.1 Dynamický zásobník

**Procvičované učivo:** dynamická alokace paměti, ukazatele, funkce, strukturované datové typy

Napište v jazyku C funkce pro práci s dynamickým zásobníkem. Připomínáme, že je o datovou strukturu, jejíž základem ke prvek, který v sobě kromě dat (při řešení této úlohy postačí jedna položka typu `int`) obsahuje také ukazatel na prvek, který byl do zásobníku vložen bezprostředně před ním. Se zásobníkem je možné provádět tyto operace: přidání prvku - funkce `prvek *pridej(prvek *zasobnik, int data)`, přečtení dat z vrchoolu zásobníku - funkce `int vrchol(prvek* zasobnik)` a odebrání prvku z vrcholu zásobníku - funkce `prvek *odeber(prvek* zasobnik)`.

**Příklad použití:**

```
int main(){
   prvek* z=NULL;
   int i;

   for (i=1; i<11; i++)
        z = pridej(z, i);

   while (z!=NULL){
        printf("%i\n", vrchol(z));
        z=odeber(z);
   }

   return 0;
}
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** dynamická fronta, dynamický seznam

## 9.2 Spojení textových řetězců

**Procvičované učivo:** dynamická alokace paměti, ukazatele, funkce

Napište v jazyku C funkci `char *spojeni(char *t1, char *t2)`, která vytvoří a vrátí textový řetězec, který vznikne spojením předaných textových řetězců. Ve funkci `main` pak tuto funkci otestujte...

**Příklad výstupu:**

```
Spojeni slov "Ahoj" a "Svete" je "AhojSvete".
```

**Povolené knihovny:** stdio.h, stdlib.h

# 10 Předání parametru odkazem

## 10.1 Transformace textu

**Procvičované učivo:** předání parametru odkazem, dynamická alokace paměti, práce s textovými řetězci, funkce

Napište v jazyku C funkci `int set(char* in, char** out)`, která podle textového řetězce `in` vytvoří řetězec, který naváže na ukazatel `out`. Vytváření výstupního textu probíhá tak, že malé písmeno je při kopírování nahrazeno odpovídajícím velkým písmenem a naopak. Jiné znaky se zkopírují bez změny. Funkce vrací počet pozměněných znaků.

**Příklad výstupu:**

```
Puvodni text: "Ahoj svete 23."
Zmeneny text: "aHOJ SVETE 23."
```

**Povolené knihovny:** stdio.h, stdlib.h

## 10.2 Celočíselné dělení

**Procvičované učivo:** předání parametru odkazem, ukazatele, funkce, cykly, aritmetické operátory

Napište v jazyku C funkci `int deleni(int a, int b, int *r)`, která podělí číslo `a` číslem `b` a vrátí podíl těchto čísel. Pomocí parametru `r` se z funkce vrací také zbytek po provedeném celočíselném dělení. Ve funkci `deleni` není dovoleno použít operátory `/` a `%`. Funkci otestujte a výsledky výpočtů vypište ve funkci `main` na obrazovku.

**Příklad výstupu:**

```
13 : 4 = 3 (zbytek 1)
```

**Povolené knihovny:** stdio.h, stdlib.h

# 11 Procvičení učiva I

## 11.1 Četnost znaků

**Procvičované učivo:** dynamická alokace paměti, práce s textem, funkce, pole, cykly

Napište v jazyku C funkci `int *cetnost(char *text)`, která určí počet výskytů jednotlivých znaků ve vstupním textovém řetězci `text`. Funkce pak vrací pole četností znaků, kde *i*-tý prvek pole odpovídá počtu výskytů *i*-tého znaku ASCII tabulky. Funkce musí rozpoznávat všechny znaky ASCII tabulky, tedy 256 znaků. Ve funkci `main` vytvořenou funkci otestujte.

**Příklad výstupu:**

```
Vypis cetnosti znaku vyskytujicich se v retezci
"aaa bb ccc ddd fffeeeshdhdkkay23455"
(znak:cetnost)

 : 4
2: 1
3: 1
4: 1
5: 2
a: 4
b: 2
c: 3
d: 5
e: 3
f: 3
h: 2
k: 2
s: 1
y: 1
```

**Povolené knihovny:** stdio.h, stdlib.h

## 11.2 Mincovka

**Procvičované učivo:** předání parametru odkazem, dynamická alokace paměti, funkce, pole, cykly

Tato úloha je založena na jednoduchém problému, který každodenně řeší pokladní v nejrůznějších obchodech. Ne vždy dokáže zákazník zaplatit zboží přesným obnosem a je tudíž třeba mu jeho "přeplatek" vrátit a to nejlépe nejmenším počtem platidel, aby se minimalizovala možnost nějakého omylu.

Vašim úkolem tedy bude napsat v jazyku C funkci `int mincovka(unsigned int castka, unsigned int **platidla)`, která tento problém řeší. Vstupním parametrem funkce je nezáporný celočíselný finanční obnos `castka`, který je třeba zákazníkovi vrátit nejmenším možným počtem platidel. Výstupem funkce je počet použitých platidel, který se vrací návratovou hodnotou, a pole hodnot těchto platidel uspořádané od největší hodnoty po nejnižší, které je vráceno pomocí parametru `platidla`. Pro úplnost dodáme, že používáme platidla s hodnotami 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000 a 5000 Kč.

**Příklad výstupu:**

```
castka: 12345
pouzita platidla: 5000, 5000, 2000, 200, 100, 20, 20, 5
```

**Povolené knihovny:** stdio.h, stdlib.h

## 11.3 Převody mezi číselnými soustavami

**Procvičované učivo:** dynamická alokace paměti, práce s textem, funkce, cykly

Napište v jazyku C funkci `char *preved(unsigned int z1, unsigned int z2, char *cislo)` pro převádění celých nezáporných čísel mezi číselnými soustavami. Vstupními parametry požadované funkce jsou základ číselné soustavy `z1`, ze které se převádí (”vstupní” číselná soustava), základ soustavy `z2`, do které se převádí (”výstupní” číselná soustava), a textový řetězec `cislo` odpovídající číslu ve ”vstupní” číselné soustavě. Výstupem funkce je textový řetězec odpovídající zadanému číslu vyjádřenému ve ”výstupní” číselné soustavě nebo konstanta NULL, pokud textový řetězec `cislo` nebyl korektním zadáním čísla ve ”vstupní” číselné soustavě (v řetězci se vyskytly jiné znaky než cifry dané soustavy).

Pro výpis číslic s hodnotami většími než 9 použijte velká písmena anglické abecedy. Předpokládejte základy číselných soustav v rozmezí od 2 do 36 a hodnotu vstupního čísla (textového řetězce) v rozsahu datového typu `unsigned long`.

**Příklad výstupu:**

```
10111 v soustave o zakladu 2 odpovida 23 v soustave o zakladu 10
1202101 v soustave o zakladu 3 odpovida 1671 v soustave o zakladu 9
26A1B800A v soustave o zakladu 12 odpovida 3F0J48I v soustave o zakladu 26
```

**Povolené knihovny:** stdio.h, stdlib.h

## 11.4 Počet výskytů řetězce v řetězci

**Procvičované učivo:** práce s textem, funkce, cykly

Napište v jazyku C funkci `int pocet_vyskytu(char *kde, char *co)`, která vypočítá a vrátí počet výskytů textového řetězce `co` v textovém řetězci `kde`. Vytvořenou funkci otestujte ve funkci `main`.

**Příklad použití:**

```
pocet = pocet_vyskytu("aalaalaa", "aalaa");
printf("Pocet vyskytu \"aalaa\" v \"aalaalaa\" je %i.\n", pocet);
```

**Příklad výstupu:**

```
Pocet vyskytu "aalaa" v "aalaalaa" je 2.
```

**Povolené knihovny:** stdio.h, stdlib.h

# 12 Rekurzivní funkce

## 12.1 Fibonacciho čísla

**Procvičované učivo:** rekurze, funkce, cykly

Napište v jazyku C rekurzivní a nerekurzivní funkci pro výpočet *n*-tého fibonacciho čísla. Porovnejte rychlosti výpočtu těchto dvou funkcí pro větší *n*.

Připomínáme, že posloupnost fibonacciho čísel je definována rekurzivně:

Fib(0) = 0

Fib(1) = 1

Fib(n) = Fib(n-1) + Fib(n-2) pro n $>$ 1

**Příklad výstupu:**

```
Fib(8)  = 21
Fib(24) = 46368
Fib(35) = 9.22747E+006
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** výpočet faktoriálu

## 12.2 Hledání půlením intervalu

**Procvičované učivo:** rekurze, funkce, pole, větvení

Napište v jazyku C rekurzivní funkci `int puleni(int cisla[], int a, int b, int hledane)`, která pomocí metody půlení intervalu najde v zadaném setříděném poli `cisla` hodnotu `hledane` a vrátí její index v tomto poli. Připomínáme, že metoda půlení intervalu je založena na porovnání hodnoty hledaného čísla s číslem "uprostřed" právě prohledávaného intervalu (v našem případě intervalu mezi prvky s indexy `a` a `b`). Pokud se hodnoty rovnají, našli jsme hledané číslo a můžeme tedy přímo vrátit jeho index. Pokud se nerovnají, stačí (rekurzivním voláním) prohledávat pouze jeden z intervalů prvek s indexem `a` až "prostřední prvek" nebo "prostřední prvek" až prvek s indexem `b`.

**Příklad použití:**

```
int main(){
   int p[11];
   int i;

   for (i=0; i<11; i++)
      p[i] = 2*i+3;

   for (i=0; i<11; i++)
      printf("Cislo %i je na indexu: %i\n", 2*i+3, puleni(p,0,10, 2*i+3));

   return 0;
}
```

**Příklad výstupu:**

```
Cislo 3 je na indexu: 0
Cislo 5 je na indexu: 1
Cislo 7 je na indexu: 2
Cislo 9 je na indexu: 3
Cislo 11 je na indexu: 4
Cislo 13 je na indexu: 5
Cislo 15 je na indexu: 6
Cislo 17 je na indexu: 7
Cislo 19 je na indexu: 8
Cislo 21 je na indexu: 9
Cislo 23 je na indexu: 10
```

**Povolené knihovny:** stdio.h, stdlib.h

# 13 Statická vícerozměrná pole

## 13.1 Maximum dvojrozměrného pole

**Procvičované učivo:** statická vícerozměrná pole, funkce, cykly

Napište v jazyku C funkci `int maximum(int prvky[][4], int radku)`, která vrátí hodnotu největšího čísla uloženého ve dvourozměrném poli `prvky`. První rozměr pole `prvky` lze určit pomocí parametru `radku`, druhý je pevně dán konstantou 4.

**Příklad výstupu:**

```
Vypis pole:
 10   2  15  -2
-52  41   0  12
 15   3   1  -8

Maximum je: 41
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** suma, průměr, minimum hodnot dvourozměrného pole

## 13.2 Suma řádků dvojrozměrného pole

**Procvičované učivo:** statická vícerozměrná pole, dynamická alokace paměti, funkce, cykly

Napište v jazyku C funkci `int *suma_radku(int prvky[][4], int radku)`, která vypočítá součty na jednotlivých řádcích pole `prvky` a vrátí jednorozměrné pole obsahující tyto součty. První rozměr pole `prvky` lze určit pomocí parametru `radku`, druhý je pevně dán konstantou 4.

**Příklad výstupu:**

```
Vypis pole:
 10   2  15  -2
-52  41   0  12
 15   3   1  -8

Soucty na radcich jsou: 25, 1, 11
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** aritmetický průměr jednotlivých řádků

# 14 Dynamická vícerozměrná pole

## 14.1 Součin matic

**Procvičované učivo:** dynamická vícerozměrná pole, funkce, cykly

Napište v jazyku C funkci `double **soucin(int m, int n, int o, double **A, double **B)`, která vypočítá součin matice `A` o rozměrech `mxn` a matice `B` o rozměrech `nxo`. Funkce vrací alokované dvojrozměrné pole s hodnotami výsledné matice.

**Příklad výstupu:**

```
Matice A:
 1  2  3
 4  5  6

Matice B:
 1  0
 2  1
 0 -1

Vysledna matice:
 5 -1
14 -1
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** součet matic

## 14.2 Četnost znaku v poli řetězců

**Procvičované učivo:** dynamická vícerozměrná pole, funkce, cykly

Napište v jazyku C funkci `int vyskyty(char* texty[], int pocet, char hledany)`, která vrací počet výskytů znaku `hledany` v poli textových řetězců `texty`. Rozměr pole textových řetězců (počet textových řetězců v poli) lze specifikovat pomocí parametru `pocet`. Pro testování funkce si v `main` funkci vytvořte libovolné pole textových řetězců.

**Příklad výstupu:**

```
Textove retezce:
Ahoj uzivateli,
jak se mas?
Tohle bude snadne, ne?

Znak "e" se v poli vyskytuje 6krát.
```

**Povolené knihovny:** stdio.h, stdlib.h

# 15 Ukazatele na funkce

## 15.1 Mapování funkce

**Procvičované učivo:** ukazatele na funkce, dynamická alokace paměti, funkce, pole

Napište v jazyku C funkci `double *map(double (*fce)(double), double *vstup, int delka)`, která na hodnoty pole `vstup` (definiční obor) aplikuje funkci `fce` a vrátí pole výsledných hodnot. Velikost definičního oboru je specifikována parametrem `delka`.

**Příklad použití:**

```
double na2(double x){
        return x*x;
}

double na3(double x){
   return x*x*x;
}

int main(){
   ...
   pole_vysledku_na2 = map(na2, vstup,5)
   ...
   pole_vysledku_na3 = map(na3, vstup,5)
   ...
}
```

**Příklad výstupu:**

```
Vstupni pole:   1,  2,  3,  4,   5
Druhe mocniny:  1,  4,  9, 16,  25
Treti mocniny:  1,  8, 27, 64, 125
```

**Povolené knihovny:** stdio.h, stdlib.h, math.h

## 15.2 Mapování pole funkcí

**Procvičované učivo:** ukazatele na funkce, dynamická alokace paměti, funkce, pole

Napište v jazyku C funkci `double **map(double(*fce[])(double), double *vstup, int pocet_fce, int pocet_vstup)`, která na prvky pole `vstup` (definiční obor) mapuje postupně jednotlivé funkce z pole `fce`. Z vypočtených hodnot vytvoří dvourozměrné pole, které bude návratovu hodnotou z této funkce. První řádek výstupního pole bude odpovídat definičnímu oboru, druhý řádek hodnotám první funkce a tak dále, až poslední řádek bude odpovídat hodnotám poslední předané funkce. Počet funkcí je specifikován parametrem `pocet_fce`, velikost definičního oboru pak parametrem `pocet_vstup`.

**Příklad použití:**

```
double na2(double x){
        return x*x;
}

double na3(double x){
   return x*x*x;
}

int main(){
   ...
   pole_fci[0] = na2;
   pole_fci[1] = na3;
   ...
   pole_vysledku = map(pole_fci, pole, 2, 5);
   ...
}
```

**Příklad výstupu:**

```
Hodnoty vystupniho pole:
1   2   3   4   5
1   4   9  16  25
1   8  27  64 125
```

**Povolené knihovny:** stdio.h, stdlib.h, math.h

## 15.3 Akumulátor

**Procvičované učivo:** ukazatele na funkce, funkce, pole, cykly, větvení

Napište v jazyku C funkci `double akumulator(double (*fce)(double, double), double cisla[], int pocet)`, která zpracuje pomocí předané funkce `fce` hodnoty z pole `cisla`, jehož velikost je dána parametrem `pocet`. Vytvořenou funkci otestujte ve funkci `main`; použitými akumulačními funkcemi mohou být například funkce pro součet nebo součin dvou reálných čísel, které je ovšem pro testování potřeba dodefinovat.

**Příklad použití:**

```
int main(){
   double p[10];
   int i;

   for (i=0; i<10; i++)
      p[i] = i+1;

   printf("Suma je: %g\n", akumulator(soucet,p,10));
   printf("Produkt je: %g\n", akumulator(soucin,p,10));

   return 0;
}
```

**Příklad výstupu:**

```
Suma je: 55
Produkt je: 3.6288e+006
```

**Povolené knihovny:** stdio.h, stdlib.h

# 16 Funkce s proměnným počtem parametrů

## 16.1 Průměr čísel

**Procvičované učivo:** funkce s proměnným počtem parametrů, cykly, větvení

Napište v jazyku C funkci `long double prumer(char* format, ...)`, která výpočítá aritmetický průměr ze zadaných hodnot různých datových typů. Typy předávaných hodnot jsou určeny pomocí parametru `format`, který může tvořit libovolná posloupnost znaků odpovídající typům následujících parametrů - znak *"i"* pro typ `int`, *"d"* pro typ `double` a *"l"* pro `long double`.

**Příklad použití:**

```
pr = prumer("idld", 1, (double)3, (long double)2, 3.0);
printf("Prumer je %Lf. \n", pr);
```

**Příklad výstupu:**

```
Prumer je 2.25.
```

**Povolené knihovny:** stdio.h, stdlib.h, stdarg.h

**Alternativy úlohy:** Suma čísel

## 16.2 Suma komplexních čísel

**Procvičované učivo:** funkce s proměnným počtem parametrů, strukturované datové typy, cykly

Napište v jazyku C funkci `komplexni suma(int pocet, ...)`, která výpočítá součet předaných komplexníxh čísel. Počet sčítaných čísel je určen pevným parametrem `pocet`, za nímž pak ve volání funkce následují hodnoty, které má funkce sčítat. Pro práci s komplexními čísly využijte vámi definovaný strukturovaný datový typ `komplexni`.

**Příklad použití:**

```
komplexni vysledek;
komplexni a = {3.1,-2.3};
komplexni b = {0.5,-3};
komplexni c = {0,1.2};

vysledek = suma(3,a,b,c);
printf("Suma je %g + %gi. \n", vysledek.realna, vysledek.imaginarni);
```

**Příklad výstupu:**

```
Suma je 3.6 + -4.1i.
```

**Povolené knihovny:** stdio.h, stdlib.h, stdarg.h

**Alternativy úlohy:** Produkt komplexních čísel

# 17 Práce s preprocesorem

## 17.1 Číslice dané soustavy

**Procvičované učivo:** makra s parametry, podmínkový operátor, logické operátory, operátory porovnání

Napište makro `je_cislice(zaklad, znak)` pro testování, zda je daný znak (určen argumentem `znak`) číslicí soustavy s daným základem (argument `zaklad`). Makro `je_cislice` by mělo korektně fungovat pro základy soustav od 2 do 36 a libovolné znaky. Pro číslice s hodnotou větší než 9 používejte pro jednoduchost pouze velká písmena anglické abecedy.

**Příklad použití:**

```
if (je_cislice(8,'8')!=0) printf("Ano\n"); else printf("Ne\n");
if (je_cislice(10+6,'0'+4)!=0) printf("Ano\n"); else printf("Ne\n");
if (je_cislice(30,'@')!=0) printf("Ano\n"); else printf("Ne\n");
```

**Příklad výstupu:**

```
Ne
Ano
Ne
```

**Povolené knihovny:** stdio.h, stdlib.h

## 17.2 Načtení čísla typu int

**Procvičované učivo:** makra s parametry, operátor čárka, základní vstup a výstup

Napište makro `cti_int(i)` pro načtení hodnoty typu `int` do proměnné `i`. Výsledná hodnota výrazu v těle makra by také měla odpovídat načtenému číslu.

Tato úloha je převzata z publikace **Pavel Herout: Učebnice jazyka C**.

**Příklad použití:**

```
int j, k;
printf("Zadejte cele cislo: ");
if ((j = cti_int(k)) == 0) printf("nula\n");
else printf("%i %i\n", j,k);
```

**Příklad výstupu:**

```
Zadejte cele cislo: 1
1 1
```

**Povolené knihovny:** stdio.h, stdlib.h

# 18 Celková koncepce programu

## 18.1 Objemy a povrchy

**Procvičované učivo:** celková koncepce programu, parametry funkce main, funkce, větvení

Napište v jazyku C program pro výpočet objemu a povrchu válce, pravidelného trojbokého, čtyřbokého a šestibokého hranolu. Parametry výpočtu by mělo být možné předávat programu při spuštění z příkazové řádky. Zdrojový kód programu by měl být rozdělen do 2 modulů. Modul hlavní funkce (soubor `main.c`) bude zajišťovat zpracování a případně načtení chybějící parametrů výpočtu, budou z něj volány funkce zajišťující vlastní výpočet a vypisovány vypočítané hodnoty na obrazovku. Druhý modul (soubory `vypocet.h` a `vypocet.c`) pak bude zajišťovat veškeré požadované výpočty. Při řešení úlohy dbejte všech zásad zmíněných na přednášce.

**Příklad použití:**

`objemy_a_povrchy.exe 0 1.2 2.4` (OS Windows)

`./objemy_a_povrchy 0 1.2 2.4` (OS Linux)

**Příklad výstupu:**

```
Valec s vyskou 1.2 a polomerem podstavy 2.4
ma povrch 54.2592 a objem 21.7037.
```

**Příklad použití:**

`objemy_a_povrchy.exe 3 2.3 4.5` (OS Windows)

`./objemy_a_povrchy 3 2.3 4.5` (OS Linux)

```
Pravidelny 3-boky hranol s vyskou 2.3 a delkou podstavne hrany 4.5
ma povrch 48.587 a objem 20.1676.
```

**Povolené knihovny:** stdio.h, stdlib.h, math.h

## 18.2 ASCII Art

**Procvičované učivo:** celková koncepce programu, dynamická práce s pamětí, funkce

Napište v jazyku C jednoduchou knihovnu funkcí pro vykreslování obrázků pomocí znaků (tzv. ASCII art).

Knihovna by měla mít tyto vlastnosti:

- Obrázky se budou vykreslovat pomocí plátna - dvojrozměrné matice, která bude obsahovat jednotlivé znaky.

- Vykreslování se tedy neprovádí přímo na výstupu, ale pouze dochází ke změně daného plátna (struktura `canvas`).

- Je možné pracovat současně s několika plátny.

- Je možné "vykreslovat" i za hranicí kreslící plochy, tyto body se ale nebudou při zobrazení plátna vykreslovat. Jinými slovy, při pokusu o kreslení mimo plátno nedojde k vyjímce při běhu programu.

- Knihovna by měla být samostatným modulem, bude tedy tvořena jedním zdrojovým a jedním hlavičkovým souborem.

V knihovně vytvořte strukturovaný datový typ `canvas` a dále definujte tyto funkce:

```
/* vytvori platno */
canvas *canvas_create(int x, int y);

/* nastavi dany bod na zadanou hodnotu */
void canvas_set_point(canvas *c, int x, int y, char character);

/* vrati znak daneho bodu */
int canvas_get_point(canvas *c, int x, int y);

/* nakresli obdelnik */
void canvas_draw_rect(canvas *c, int x, int y, int width, int height, char ch);

/* vycisti platno */
void canvas_clear(canvas *c);

/* vykresli obsah platna na standardni vystup */
void canvas_print(canvas *c);

/* vykresli obsah platna do souboru */
void canvas_output(canvas *c, FILE *f);
```

Jednotlivé funkce ve vytvořené knihovně poté otestujte z modulu hlavní funkce.

**Příklad výstupu:**

```
xxxxxxx
x     x
x   ***********
x   * x       *
x   * x       *
x   * x       *
x   ***********
x     x
x     x
xxxxxxx
```

**Příklad výstupu:**

```
 ***********
 *         *
 *  o   o  *
 *    |    *
 *         *
 *  \___/  *
 *         *
 ***********
```

**Povolené knihovny:** stdio.h, stdlib.h

# 19 Práce s textovými soubory

## 19.1 Součty řádků

**Procvičované učivo:** práce s textovými soubory, funkce, cykly

Napište v jazyku C funkci `int soucty(const char *vstup, const char *vystup)`, která čte ze vstupního souboru `vstup` desetinná čísla, počítá součty na jednotlivých řádcích a zapisuje je do výstupního souboru `vystup`. Na konec výstupního souboru pak navíc vloží sumu všech čísel ve vstupním souboru.

**Příklad vstupního souboru:**

```
    7.134     0.5198    2.436     0.9626
    1.27      1.324     0.9639    1.538     0.4995
    1.503     4.95      0.3466
    0.454     2.367     0.6877    9.057     0.1807      1.112
    4.287     8.675     1.511     0.4296    0.2331
```

**Příklad výstupního souboru:**

```
11.0524
5.5954
6.7996
13.8584
15.1357

Suma: 52.4415
```

**Povolené knihovny:** stdio.h, stdlib.h

## 19.2 Součty zlomků

**Procvičované učivo:** práce s textovými soubory, strukturované datové typy, funkce, cykly

Napište v jazyku C funkci `zlomek soucet(const char *vstup)`, která čte ze vstupního textového souboru `vstup` zlomky (resp. dvojice celých čísel), vypočítává a vrací součet všech zlomků zapsaných ve vstupním souboru. Pro snadnější práci se zlomky si definujte strukturovaný typ `zlomek`. Výsledný zlomek by měl být upraven do základního tvaru. Vykrácení zlomku do základního tvaru docílíte vydělením čitatele i jmenovatele jejich největším společným dělitelem, který můžete určit například použitím **Euklidova algoritmu**.

**Příklad vstupního souboru:**

```
1 /-8
-14/ 9
1/2
1        /     2
```

**Příklad výstupu:**

```
Soucet je: -49 / 72
```

**Povolené knihovny:** stdio.h, stdlib.h

**Alternativy úlohy:** produkt zlomků, suma nebo produkt komplexních čísel

# 20 Práce s binárními soubory

## 20.1 Jednotkové vektory

**Procvičované učivo:** práce s binárními soubory, porozumění cizímu programu, funkce, cykly, větvení

Prostudujte si zdrojový kód v připraveném souboru (příloha A) a dopište funkci `int uprav_data (char *nazev)`. Tato funkce by měla číst vektory - trojice čísel (používejte definovanou konstantu `DIMENZE`) typu `double` z binárního souboru `nazev`. Pro každý přečtený vektor funkce vypočítá jemu odpovídající vektor o jednotkové velikosti (směr a orientace vektoru zůstanou zachovány) a upraveným vektorem přepíše stará data v souboru. Funkce poté pokračuje čtením dalšího vektoru, dokud není celý obsah datového souboru zpracován.

**Povolené knihovny:** stdio.h, stdlib.h, time.h, math.h

## 20.2 Databáze osob

**Procvičované učivo:** práce s binárními soubory, porozumění cizímu programu, funkce s proměnným počtem parametrů, cykly, větvení

Prostudujte si zdrojový kód v připraveném souboru (příloha B) a dopište funkci `int vyhledej (char* soubor, char *kriteria, ...)`. Tato funkce by měla vyhledat v binární databázi `soubor` všechny osoby odpovídající daným vlastnostem a vypsat je pomocí funkce `void vypis (osoba o)` na obrazovku. Vlastnosti osob, které budou při vyhledávání zkoumány, určují jednotlivé znaky řetězce `kriteria` (znak *"j"* pro jméno osoby, *"p"* pro příjmení, *"n"* pro datum narození, *"d"* pro den narození, *"m"* pro měsíc narození, *"r"* pro rok narození, *"P"* pro pohlaví a *"s"* pro stav), za kterým pak následují vyhledávané hodnoty těchto vlastností.

**Příklad použití:**

```
vyhledej("databaze.dat", "jps", "Anna", "Novotna", VDOVEC);
```

Vypíše všechny vdovy se jménem Anna, přijmením Novotna ze souboru "databaze.dat".

**Povolené knihovny:** stdio.h, stdlib.h, string.h, time.h, stdarg.h

# 21 Bitové operátory a bitová pole

## 21.1 Datumy

**Procvičované učivo:** bitové operátory, práce s binárními soubory, union, funkce

Prostudujte si zdrojový kód v připraveném souboru (příloha C) a dopište funkci `DATUM maximum (char *nazev)`. Tato funkce by měla číst datumy (využijte bitové pole `DATUM`) z binárního souboru `nazev` a vrátit největší (nejpozdější) datum jako svou návratovou hodnotu.

Typ `DATUM` můžete také předefinovat jako union výše zmíněného bitového pole a neznáménkového celého čísla, čímž si můžete zjednodušit porovnávání 3 hodnot (rok, měsíc a den) na porovnávání jediné hodnoty. Pozor ovšem na pořadí bitů ve struktuře bitového pole a velikosti typů na konkrétním počítači. Toto řešení je implementačně závislé!

**Příklad výstupu:**

```
Nejpozdejsi datum je: 23. 4. 1995
```

**Povolené knihovny:** stdio.h, stdlib.h, time.h

## 21.2 Množinové operace

**Procvičované učivo:** bitové operátory, dynamická alokace paněti, strukturované datové typy, funkce, pole

Prostudujte si zdrojový kód v připraveném souboru (příloha D) a dopište funkce `mnozina prunik (mnozina A, mnozina B)`, `mnozina sjednoceni(mnozina A, mnozina B)` a `mnozina rozdil(mnozina A, mnozina B)` pro operace s danými množinami `A` a `B`. Pro reprezentaci množin použijte připravený strukturovaný typ `mnozina`, jehož člen `pocet` odpovídá počtu možných indexů (tj. počtu prvků univerza) a člen `prvky` obsahuje posloupnost bitů (rozčleněnou do několika položek typu `int` tvořících pole), která udává, který prvek patří (bit s hodnotou 1) a který nepatří (hodnota 0) do dané množiny.

**Příklad výstupu:**

```
Mnozina A:
3, 5, 32, 33, 34, 36, 37, 69,
Mnozina B:
2, 32, 34, 35, 36, 37, 38, 40, 42, 44, 65,
Prunik mnozin A a B:
32, 34, 36, 37,
Sjednoceni mnozin A a B:
2, 3, 5, 32, 33, 34, 35, 36, 37, 38, 40, 42, 44, 65, 69,
Rozdil mnozin A a B:
3, 5, 33, 69,
```

**Povolené knihovny:** stdio.h, stdlib.h

# 22 Procvičení učiva II

## 22.1 Maticová kalkulačka

**Procvičované učivo:** práce s textovým souborem, celková koncepce programu, dynamická vícerozměrná pole, strukturované datové typy

Napište v jazyku C program pro výpočty libovolných formulí složených z matic (velká písmena anglické abecedy), znamének plus a krát (pro operace sčítání a násobení matic) a závorek. Program bude zadání výpočtu (formuli i matice) načítat ze vstupního textového souboru a výslednou matici ukládat do výstupního textového souboru. Jména vstupního a výstupního souboru (případně cestu k souborům) bude možné specifikovat přímo při spuštění programu z příkazové řádky.

V programu by také měla být zabudována kontrola správnosti načteného zadání a ošetření možných chyb (při otevírání souboru, alokaci paměti a podobně). Program by měl být napsaný přehledně a efektivně - s důrazem na smysluplné rozdělení do několika funkcí a minimum globálně deklarovaných proměnných.

**Příklad použití:**

`./kalkulacka zadani.txt vysledek.txt` (OS Linux)

`kalkulacka.exe zadani.txt vysledek.txt` (OS Windows)

**Příklad vstupního souboru:**

```
(A * B)+(C*(A*B))

2 3
1.1 1 2
4 5 6

3 4
1 2 3 4
5 6 7 8
9 0 1 2

2 2
1 2
3 4
```

**Příklad výstupního souboru:**

```
2 4
214.2  92.4 130.6 168.8
487.3 214.6 301.9 389.2
```

**Povolené knihovny:** stdio.h, stdlib.h, string.h

## 22.2 Hledání nejdelších slov

**Procvičované učivo:** práce s textovým souborem, celková koncepce programu, vícerozměrná pole, dynamická práce s pamětí, strukturované datové typy

Napište v jazyku C program, který v daném textovém souboru vyhledá zadaný počet nejdelších slov. Název vstupního souboru a požadovaný počet hledaných nejdelších slov by mělo být možné specifikovat z příkazové řádky při spuštění programu. Nalezená nejdelší slova vypište pro jednoduchost na

obrazovku, přičemž slova se mohou opakovat (pokud se ve zpracovávaném textovém souboru vyskytují vícekrát), slova stejné délky dále třiďte podle pořadí jejich výskytu v textu.

**Příklad použití:**

`./nejdelsi_slova vstup.txt 10` (OS Linux)

`nejdelsi_slova.exe vstup.txt 10` (OS Windows)

**Příklad vstupního souboru:**

```
Prijde informatik na prijimacky na ekonomku a tam se ho ptaji:
- "Tak teda kolik je 5 + 3?"
- "8, pohotove odpovi."
- "Spravne. Dame tezsi priklad. Kolik je 7 + 4?"
- "11."
- "Vyborne. A ted nejtezsi úloha. Kolik je 2 - 3?"
- "255."

Proc si informatici pletou Halloween a Vanoce?
Protoze OCT 31 je to same co DEC 25.

Lide se deli do 10 skupin. Jedni dvojkovou soustavu znaji a druzi ne.

Lide se deli do 10 skupin. Jedni znaji dvojkovou a trojkovou soustavu,
druzi neznaji ani jednu a treti si mysleli, ze tohle je vtip
o dvojkove soustave.
```

**Příklad výstupu:**

```
informatici
informatik
prijimacky
Halloween
dvojkovou
dvojkovou
trojkovou
ekonomku
pohotove
nejtezsi
```

**Povolené knihovny:** stdio.h, stdlib.h, string.h, ctype.h

## 22.3 Medián čísel v souboru

**Procvičované učivo:** práce s textovým souborem, celková koncepce programu, dynamická práce s pamětí, pole, strukturované datové typy

Napište v jazyku C program, který načte celá čísla z daného textovém souboru a vypíše na obrazovku medián těchto hodnot. Název zpracovávaného souboru by mělo být možné určit z příkazové řádky při spouštění programu. Připomínáme, že pro nalezení mediánu daného souboru stačí hodnoty seřadit podle velikosti a vzít hodnotu, která se nalézá uprostřed. Pokud má soubor sudý počet prvků, obvykle se za medián označuje aritmetický průměr dvou "prostředních" hodnot.

**Příklad použití:**

`./median vstup.txt` (OS Linux)

`median.exe vstup.txt` (OS Windows)

**Příklad vstupního souboru:**

```
-8   8   7  -6   5
 5   2   2   3   1
 4   0   7
```

**Příklad výstupu:**

```
Median souboru je: 3
```

**Povolené knihovny:** stdio.h, stdlib.h

# A Zdrojový kód k úloze „Jednotkové vektory”

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <time.h>
4   #include <math.h>
5
6   #define VELIKOST_BLOKU 50
7   #define PRESNOST 100
8   #define DIMENZE 3
9
10  /* hlavicky lokalnich funkci */
11  int generuj_data(char *nazev, int max, long pocet);
12  int precti_data(char *nazev);
13  double vel_vektor(double *v);
14  int uprav_data(char *nazev);
15
16  /* definice funkci */
17  int main()
18  {
19      generuj_data("data.dat", 100, 123);
20      precti_data("data.dat");
21      uprav_data("data.dat");
22      printf("\nNova data: \n");
23      precti_data("data.dat");
24      system("pause");
25      return 0;
26  }
27
28  /* funkce vytvori binarni soubor s cisly typu double od 0 do max - 1,
29   * parametr pocet udava pocet cisel v souboru
30   */
31  int generuj_data(char *nazev, int max, long pocet)
32  {
33      FILE *f;
34      double data[VELIKOST_BLOKU];
35      long i, j;
36      unsigned int zapsano;
37
38      /* otevreni souboru */
39      f = fopen(nazev, "wb");
40      if (f == NULL)
41          return 1;
42
43      /* inicializace generetoru nahodnych cisel */
44      srand((unsigned) time(NULL));
45
46      /* generovani a zapis uplnych datovych bloku */
47      for (i = 0; i < pocet / VELIKOST_BLOKU; i++) {
48
49          /* generovani jednoho datoveho bloku */
50          for (j = 0; j < VELIKOST_BLOKU; j++) {
51              data[j] = (rand() % (max * PRESNOST)) / (double) PRESNOST;
52          }
```

```
53
54          /* zapis jednoho datoveho bloku */
55          zapsano = (unsigned) fwrite(data, sizeof(double), VELIKOST_BLOKU, f);
56
57          /* test uspesnosti zapisu */
58          if (zapsano != VELIKOST_BLOKU) {
59              fclose(f);
60              return 2;
61          }
62      }
63
64      /* generovani posledniho neuplneho datoveho bloku */
65      for (j = 0; j < pocet % VELIKOST_BLOKU; j++) {
66          data[j] = (rand() % (max * PRESNOST)) / (double) PRESNOST;
67      }
68
69      /* zapis posledniho datoveho bloku */
70      zapsano = (unsigned) fwrite(data, sizeof(double), pocet % VELIKOST_BLOKU, f);
71
72      /* test uspesnosti zapisu */
73      if (zapsano != pocet % VELIKOST_BLOKU) {
74          fclose(f);
75          return 2;
76      }
77
78      /* uzavreni proudu a test uspesnosti */
79      if (EOF == fclose(f))
80          return 3;
81      return 0;
82  }
83
84  /* funkce cte binarni soubor s cisly typu double a vypisuje je na std. vystup */
85  int precti_data(char *nazev)
86  {
87      FILE *f;
88      double data[VELIKOST_BLOKU];
89      unsigned int i;
90      unsigned int precteno;
91
92      /* otevreni souboru */
93      f = fopen(nazev, "rb");
94      if (f == NULL)
95          return 1;
96
97      /* cte se dokud to jde */
98      do {
99          /* cteni celeho bloku */
100         precteno = (unsigned) fread(data, sizeof(double), VELIKOST_BLOKU, f);
101
102         /* vypis bloku dat na obrazovku */
103         for (i = 0; i < precteno; i++) {
104             printf("%g ", data[i]);
105         }
106         printf("\n");
```

```
107         } while (precteno == VELIKOST_BLOKU);
108 
109         /* uzavreni proudu a test uspesnosti */
110         if (EOF == fclose(f))
111             return 3;
112         return 0;
113     }
114 
115     /* funkce pro vypocet delky vektoru urcite DIMENZE */
116     double vel_vektor(double *v)
117     {
118         double out = 0.0;
119         int i;
120         for (i = 0; i < DIMENZE; i++)
121             out += v[i] * v[i];
122         return sqrt(out);
123     }
124 
125     int uprav_data(char *nazev)
126     {
127         /* DOPLNTE */
128     }
```

# B Zdrojový kód k úloze „Databáze osob”

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <time.h>
5  #include <stdarg.h>
6
7  #define MAX_ROK 2010
8  #define MIN_ROK 1950
9  #define POCET_STAVU 4
10 #define DELKA 20
11
12 const int jmena_m_pocet = 10;
13 const int jmena_z_pocet = 10;
14 const int prijmeni_m_pocet = 10;
15 const int prijmeni_z_pocet = 10;
16 const char *jmena_m[] =
17     { "Jakub", "Jan", "Tomas", "Lukas", "Ondrej", "Vojtech", "Matej",
18 "Adam", "Daniel", "Filip" };
19 const char *jmena_z[] =
20     { "Tereza", "Natalie", "Anna", "Adela", "Eliska", "Karolina",
21 "Katerina", "Barbora", "Lucie", "Kristyna" };
22 const char *prijmeni_m[] =
23     { "Novak", "Svoboda", "Novotny", "Dvorak", "Cerny", "Prochazka",
24 "Kucera", "Vesely", "Horak", "Nemec" };
25 const char *prijmeni_z[] =
26     { "Novakova", "Svobodova", "Novotna", "Dvorakova", "Cerna",
27 "Prochazkova", "Kucerova", "Vesela", "Horakova", "Nemcova" };
28
29 typedef enum {
30     MUZ = 'M', ZENA = 'Z'
31 } POHLAVI;
32
33 typedef enum {
34     SVOBODNY = 'S', ZENATY = 'Z', ROZVEDENY = 'R', VDOVEC = 'V'
35 } STAV;
36
37 typedef struct {
38     char den;
39     char mesic;
40     int rok;
41 } datum;
42
43 typedef struct {
44     char jmeno[DELKA];
45     char prijmeni[DELKA];
46     datum narozen;
47     POHLAVI pohlavi;
48     STAV stav;
49 } osoba;
50
51 int je_prestupny(unsigned int rok);
52 unsigned int pocet_dnu(unsigned int mesic);
```

```
53  datum generuj_datum(int rok_od, int rok_do);
54  int generator(char *soubor, int pocet);
55  int vyhledej(char *soubor, char *kriteria, ...);
56  void vypis(osoba o);
57
58  int main()
59  {
60      /* vygenerovani binarni databaze */
61      generator("databaze.dat", 1000);
62
63      /* vyhledavani v databazi */
64      vyhledej("databaze.dat", "P", ZENA);
65      printf("\n");
66      vyhledej("databaze.dat", "jps", "Anna", "Novotna", VDOVEC);
67
68      return 0;
69  }
70
71  int je_prestupny(unsigned int rok)
72  {
73      if (rok % 100 == 0)
74          return (rok % 400 == 0);
75      else
76          return (rok % 4 == 0);
77  }
78
79  unsigned int pocet_dnu(unsigned int mesic)
80  {
81      const static int pocet[] =
82          { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
83      if (mesic <= 12)
84          return pocet[mesic - 1];
85      else
86          return 0;
87  }
88
89  datum generuj_datum(int rok_od, int rok_do)
90  {
91      char den;
92      char mesic;
93      int rok;
94      char max_den;
95      datum d;
96
97      rok = (rand() % (rok_do - rok_od + 1)) + rok_od;
98      mesic = (rand() % 12) + 1;
99      max_den = pocet_dnu(mesic);
100     if ((mesic == 2) && (je_prestupny(rok)))
101         max_den++;
102     den = (rand() % max_den) + 1;
103
104     d.den = den;
105     d.mesic = mesic;
106     d.rok = rok;
```

```
107         return d;
108     }
109
110     int generator(char *soubor, int pocet)
111     {
112         int i;
113         size_t zapsano;
114         FILE *fw;
115
116         /* otevreni souboru */
117         fw = fopen(soubor, "wb");
118         if (fw == NULL) {
119             printf("Chyba: Vystupni soubor nebyl vytvoren. \n");
120             return 1;
121         }
122
123         /* generovani udaju a zapis do souboru */
124         srand((unsigned) time(NULL));
125         for (i = 0; i < pocet; i++) {
126             osoba o;
127             char stav_cislo;
128
129             o.pohlavi = (rand() % 2) ? MUZ : ZENA;
130             if (o.pohlavi == MUZ) {
131                 strcpy(o.jmeno, jmena_m[rand() % jmena_m_pocet]);
132                 strcpy(o.prijmeni, prijmeni_m[rand() % prijmeni_m_pocet]);
133             } else {
134                 strcpy(o.jmeno, jmena_z[rand() % jmena_z_pocet]);
135                 strcpy(o.prijmeni, prijmeni_z[rand() % prijmeni_z_pocet]);
136             }
137             o.narozen = generuj_datum(MIN_ROK, MAX_ROK);
138             stav_cislo = rand() % POCET_STAVU;
139             switch (stav_cislo) {
140             case 0:
141                 o.stav = SVOBODNY;
142                 break;
143             case 1:
144                 o.stav = ZENATY;
145                 break;
146             case 2:
147                 o.stav = ROZVEDENY;
148                 break;
149             case 3:
150                 o.stav = VDOVEC;
151                 break;
152             }
153
154             /* zapis a test */
155             zapsano = fwrite(&o, sizeof(osoba), 1, fw);
156             if (zapsano != 1) {
157                 fclose(fw);
158                 return 2;
159             }
160         }
```

```
161         /* uzavreni souboru a test */
162         if (fclose(fw) == EOF) {
163             printf("Chyba: Vystupni soubor nebyl uzavren. \n");
164             return 3;
165         }
166 
167         return 0;
168     }
169 
170     int vyhledej(char *soubor, char *kriteria, ...)
171     {
172         /* DOPLNTE */
173     }
174 
175     void vypis(osoba o)
176     {
177         static char *stavy[] =
178             { "svobodny", "svobodna", "zenaty", "vdana", "rozvedeny",
179     "rozvedena", "vdovec", "vdova" };
180         int stavy_index;
181 
182         printf("%s %s ", o.jmeno, o.prijmeni);
183 
184         if (o.pohlavi == MUZ)
185             printf("narozen ");
186         else
187             printf("narozena ");
188 
189         printf("%i. %i. %i", o.narozen.den, o.narozen.mesic, o.narozen.rok);
190 
191         switch (o.stav) {
192         case SVOBODNY:
193             stavy_index = 0;
194             break;
195         case ZENATY:
196             stavy_index = 2;
197             break;
198         case ROZVEDENY:
199             stavy_index = 4;
200             break;
201         case VDOVEC:
202             stavy_index = 6;
203             break;
204         }
205         stavy_index += (o.pohlavi == MUZ) ? 0 : 1;
206         printf(", %s\n", stavy[stavy_index]);
207 
208     }
```

# C Zdrojový kód k úloze „Datumy”

```
1
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <time.h>
5
6   #define MIN_ROK 1980
7   #define MAX_ROK 2000
8
9   typedef struct {
10      unsigned den:5;
11      unsigned mesic:4;
12      unsigned rok:7;
13  } DATUM;
14
15  int je_prestupny(unsigned int rok);
16  unsigned int pocet_dnu(unsigned int mesic);
17  int generuj_datumy(const char *nazev, unsigned int pocet);
18  int precti_datumy(char *nazev);
19  DATUM maximum(char *nazev);
20
21  int main()
22  {
23      DATUM m;
24      generuj_datumy("datumy.dat", 100);
25      precti_datumy("datumy.dat");
26      m = maximum("datumy.dat");
27      printf("\nNejpozdejsi datum je: %u. %u. %u\n",
28             m.den, m.mesic, m.rok + MIN_ROK);
29      system("pause");
30      return 0;
31  }
32
33  int je_prestupny(unsigned int rok)
34  {
35      if (rok % 100 == 0)
36          return (rok % 400 == 0);
37      else
38          return (rok % 4 == 0);
39  }
40
41  unsigned int pocet_dnu(unsigned int mesic)
42  {
43      const static int pocet[] = { 31, 28, 31, 30, 31, 30,
44          31, 31, 30, 31, 30, 31
45      };
46      if (mesic <= 12)
47          return pocet[mesic - 1];
48      else
49          return 0;
50  }
51
52  int generuj_datumy(const char *nazev, unsigned int pocet)
```

```
53  {
54      unsigned int i;
55      size_t zapsano;
56      FILE *fw;
57
58      fw = fopen(nazev, "wb");
59      if (fw == NULL)
60          return 1;
61
62      srand((unsigned) time(NULL));
63      for (i = 0; i < pocet; i++) {
64          unsigned int den;
65          unsigned int mesic;
66          unsigned int rok;
67          unsigned int max_den;
68          DATUM d;
69          rok = (rand() % (MAX_ROK - MIN_ROK + 1)) + MIN_ROK;
70          mesic = (rand() % 12) + 1;
71          max_den = pocet_dnu(mesic);
72          if ((mesic == 2) && (je_prestupny(rok)))
73              max_den++;
74          den = (rand() % max_den) + 1;
75
76          d.den = den;
77          d.mesic = mesic;
78          d.rok = rok - MIN_ROK;
79          zapsano = fwrite(&d, sizeof(DATUM), 1, fw);
80          if (zapsano != 1) {
81              fclose(fw);
82              return 2;
83          }
84      }
85      if (EOF == fclose(fw))
86          return 3;
87      return 0;
88  }
89
90  int precti_datumy(char *nazev)
91  {
92      FILE *fr;
93      DATUM d;
94      size_t precteno;
95
96      fr = fopen(nazev, "rb");
97      if (fr == NULL)
98          return 1;
99
100     /* cte dokud to jde - neni testovano,
101      * jestli skonci na konci souboru nebo nastane chyba
102      */
103     do {
104         precteno = fread(&d, sizeof(DATUM), 1, fr);
105         printf("%u. %u. %u\n", d.den, d.mesic, d.rok + MIN_ROK);
106     } while (precteno == 1);
```

```
107
108         if (EOF == fclose(fr))
109             return 3;
110         return 0;
111     }
112
113     DATUM maximum(char *nazev)
114     {
115         /* DODELAT */
116     }
```

# D Zdrojový kód k úloze „Množinové operace”

```
1
2   #include <stdio.h>
3   #include <stdlib.h>
4
5   /* struktura reprezentujici mnozinu */
6   typedef struct {
7       int *prvky;                 /* pole pro bitove ulozeni prvku */
8       int pocet;                  /* pocet prvku v mnozine */
9   } mnozina;
10
11  void vypis(mnozina A);          /* vypis indexu prvku dane mnoziny na obrazovku */
12  mnozina prunik(mnozina A, mnozina B);   /* prunik dvou mnozin */
13  mnozina sjednoceni(mnozina A, mnozina B);       /* sjednoceni dvou mnozin */
14  mnozina rozdil(mnozina A, mnozina B);   /* rozdil prvni a druhe mnoziny */
15
16  int main()
17  {
18      mnozina A, B, C;
19
20      /* vytvoreni mnoziny A */
21      A.prvky = (int *) malloc(3 * sizeof(int));
22      A.prvky[0] = 43;
23      A.prvky[1] = 55;
24      A.prvky[2] = 42351;
25      A.pocet = 3 * sizeof(int) * 8;
26
27      /* vytvoreni mnoziny B */
28      B.prvky = (int *) malloc(3 * sizeof(int));
29      B.prvky[0] = 43;
30      B.prvky[1] = 5501;
31      B.prvky[2] = 42000;
32      B.pocet = 3 * sizeof(int) * 8;
33
34      /* vypis mnoziny A na obrazovku */
35      printf("Mnozina A: \n");
36      vypis(A);
37      printf("\n");
38
39      /* vypis mnoziny B na obrazovku */
40      printf("Mnozina B: \n");
41      vypis(B);
42      printf("\n");
43
44      /* vypocet pruniku a jeho vypis na obrazovku */
45      C = prunik(A, B);
46      printf("Prunik mnozin A a B: \n");
47      vypis(C);
48      printf("\n");
49
50      /* vypocet sjednoceni a jeho vypis na obrazovku */
51      free(C.prvky);
52      C = sjednoceni(A, B);
```

```
53      printf("Sjednoceni mnozin A a B: \n");
54      vypis(C);
55      printf("\n");
56
57      /* vypocet rozdilu a jeho vypis na obrazovku */
58      free(C.prvky);
59      C = rozdil(A, B);
60      printf("Rozdil mnozin A a B: \n");
61      vypis(C);
62      printf("\n");
63
64      system("pause");
65      return 0;
66  }
67
68  void vypis(mnozina A)
69  {
70      long maska;
71      int index;
72      int i, j;
73      int mez_i;
74
75      index = 0;
76
77      /* vypocet meze cyklu */
78      mez_i = A.pocet / (sizeof(int) * 8) + 1;
79
80      /* cyklus pres cisla typu int */
81      for (i = 0; i < mez_i; i++) {
82
83          /* vypocet meze cyklu */
84          int mez_j = (i < mez_i - 1) ? sizeof(int) * 8
85              : (A.pocet % (sizeof(int) * 8));
86
87          /* posun masky na zacatek */
88          maska = 1;
89
90          /* cyklus pres jednotlive bity */
91          for (j = 0; j < mez_j; j++) {
92
93              /* pokud je prvek v mnozine - bit 1 pod maskou,
94               * tak vypiseme odpovidajici index
95               */
96              if (A.prvky[i] & maska)
97                  printf("%i, ", index);
98              maska *= 2;         /* posun masky */
99              index++;            /* posun indexu */
100         }
101     }
102 }
103
104 mnozina prunik(mnozina A, mnozina B)
105 {
106     /* DOPLNTE */
```

```
107 }
108
109 mnozina sjednoceni(mnozina A, mnozina B)
110 {
111     /* DOPLNTE */
112 }
113
114 mnozina rozdil(mnozina A, mnozina B)
115 {
116     /* DOPLNTE */
117 }
```

# Literatura


[1] Pavel Herout: *Učebnice jazyka C.* Kopp, 2007, ISBN 80-7232-220-6.

[2] Pavel Herout: *Učebnice jazyka C, 2.* díl. Kopp, 2006, ISBN 80-7232-221-4.

[3] Reek Kenneth: *Pointers on C.* Addison Wesley, 1997, ISBN: 978-0673999863.

[4] Brian W. Kernighan, Dennis M. Ritchie: *Programovací jazyk C.* Computer Press, 2008, ISBN 80-251-0897-X.

[5] Robert Sedgewick: *Algorithms in C.* Addison-Wesley Professional, 2001, ISBN: 978-0201756081.

[6] Jeri R. Hanly, Elliot B. Koffman: *Problem Solving and Program Design in C.* Addison Wesley, 2006, ISBN: 978-0321409911.

[7] Eric S. Roberts: *Programming Abstractions in C.* Addison Wesley, 1997, ISBN: 978-0201545418.

[8] Eric S. Roberts: *The Art and Science of C.* Addison Wesley, 1994, ISBN:978-0201543223.