Sbírka příkladů z jazyka C

Jan Fikejz

Recenzenti:
Ing. Jan Panuš, Ph.D.
Ing. Michal Varga, PhD.

Tato e-kniha neprošla jazykovou korekturou.

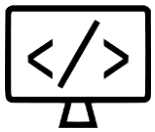
Anotace

Tato cvičebnice si klade za primární cíl poskytnout doplňkovou studijní oporu ke studiu a přípravě ke zkoušce z předmětu **Jazyk C**, který je povinný pro posluchače prezenčního studia bakalářského programu **Informační technologie** na Fakultě elektrotechniky a informatiky Univerzity Pardubice.

Obsah

1. Základní pojmy a definice jazyka C

V této části se zaměříme na procvičení základních pojmů a definic jazyka C. Dále je pozornost věnována na obecné procvičení algoritmizace pomocí vývojových diagramů.



Příklad 1.1

Navrhněte vývojový diagram algoritmu, který provede součet 10 vstupních hodnot, které jsou reprezentovány proměnnými **`a1`** až **`a10`**.

Příklad 1.2

Navrhněte vývojový diagram algoritmu, který určí kolik z 10 vstupních hodnot je kladných. Vstupní hodnoty jsou reprezentovány proměnnými `a1` až `a10`. Výsledný počet kladných hodnot bude uložen v proměnné `pocetKladnych`.

Příklad 1.3

Navrhněte vývojový diagram algoritmu, který vypočítá aritmetický průměr z posloupnosti celých kladných čísel. Konec posloupnosti je signalizován hodnotou nula, která do posloupnosti nepatří.

Příklad 1.4

Navrhněte vývojový diagram algoritmu, který vypočítá obvod kruhu. Poloměr je zadán uživatelem a je uložen do proměnné R.

Příklad 1.5

Navrhněte vývojový diagram algoritmu nápojového automatu. Automat nabízí tři druhy nápojů. Nápoj A za 10, nápoj B za 14 a nápoj C za 16. Uživatel nejdříve zvolí požadovaný nápoj. Následně jsou vhazovány různé mince, dokud automat nedisponuje částku rovnou nebo větší než cena nápoje. Automat přebytek peněz vrátí zpět.

Příklad 1.6

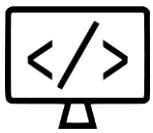
Navrhněte vývojový diagram algoritmu, který čte ze vstupního souboru jednotlivé znaky. Pokud je znak malé písmeno, je převeden na velké písmeno znak a zapsán do výstupního souboru. Znak je zapsán do výstupního souboru bez změny.

Příklad 1.7

Navrhněte vývojový diagram algoritmu bankomatu, který vydává peníze. Po vložení karty je uživatel ověřen (max. 3 pokusy, pak následuje blokace karty). Dále uživatel zadá požadovaný výběr. Požadovaná částka je zkontrolována bankomatem (min požadavek 200 Kč max požadavek 20.000 Kč). Dále je zkontrolována výše zůstatku uživatele na jeho účtu. Pokud je na účtu dostatečný zůstatek, je částka vyplacena.

2. Definice proměnných, operátory, přiřazení, přetypování, typedef, ternární operátor

Tento blok se věnuje praktickému procvičení problematiky definice proměnných, a to z pohledu jejich definice, respektive deklarace. Dále jsou procvičeny operátory a přetypování.



Příklad 2.1

Napište program v jazyku C, který vypočítá obsah a obvod obdélníka a vypíše jej na obrazovku. Vstupem programu jsou strany tohoto tělesa.

```
Zadejte stranu a: 8
Zadejte stranu b: 5
Obsah obdelniku je: 40
Obvod obdelniku je: 26
```

Příklad 2.2

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

```
Zadejte cislo: 6.124
Cela cast cisla 6.124 je 6
```

Příklad 2.3

Napište v jazyku C program, který načte desetinné číslo. Dále načte požadovanou přesnost a vstupní hodnotu zaokrouhlí dle této přesnosti.

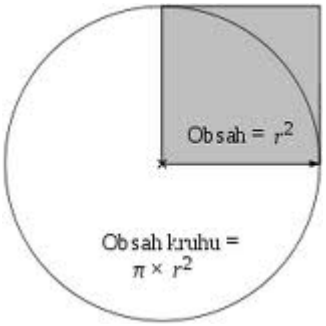
```
Zadejte cislo: 6.127
Zadejte pozadovanou presnost: 0.01
Zaokrouhlene cislo je: 6.13
```

Příklad 2.4

Napište v jazyku C program, který vypočítá:

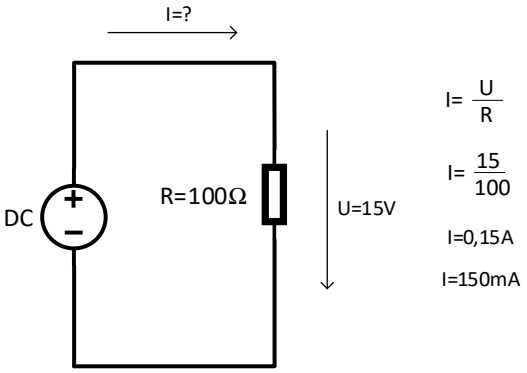
- objem a povrch zeměkoule,
- obsah a obvod jejího řezu (kružnice) v nejširším místě.

Výsledek vypište na obrazovku.



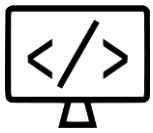
Příklad 2.5

Napište v jazyku C program, který umožňuje vypočítat proud procházející elektrickým obvodem, pomocí Ohmova zákona



3. Řídicí struktury, základní vstupy a výstupy

Tento blok se věnuje praktickému procvičení problematiky řídicích struktur jazyka C, jako jsou cykly a větvení programu. Dále je procvičena základní práce se vstupy a výstupy.



Příklad 3.1

Napište program v jazyku C, který z klávesnice načte strany trojúhelníka a, b, c, a následně provede test, zda jde sestrojit (a+b>c, a+c>b, b+c>a). Pokud lze trojúhelník sestrojit, pak vypište jeho obvod, v opačném případě oznamte uživateli, že takový trojúhelník sestrojit nelze.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadejte stranu a: 8
Zadejte stranu b: 5
Zadejte stranu c: 6
Trojuhelnik lze sestrojit a jeho obvod je 19
```

Příklad 3.2

Napište program v jazyku C, který nabídne uživateli 3 varianty výpisu 10 hodnot (FORem, WHILEem, DO-WHILEem) v případě, že volba neodpovídá, vypíše se oznámení, že zadaná volba neodpovídá nabídce.

```
Vypis cyklem FOR:
0
1
2
3
4
5
6
7
8
9
```

Příklad 3.3

Rozšiřte předchozí program o další smyčku a volbu KONEC, který ukončí smyčku menu.

Příklad 3.4

Napište program v jazyku C, který počítá počet zadaných velkých a malých znaků a/A . Program běží v nekonečné smyčce a v každé iteraci je vypsán aktuální stav.

```
Zadej znak: a

pocet a: 6
pocet A: 2
ostani znaky: 10
```

Příklad 3.5

Napište program v jazyku C, který počítá progresivní daň ze mzdy podle následujících pravidel:

1. 10% do 10.000 Kč
2. 20% do 20.000 Kč
3. 30% nad 20.000 Kč

Daň se vypočítá tak, že každá část je daněna příslušným procentem. Tedy u mzdy 35600 se vypočte následovně. Prvních 10.000 (10%), dalších 20.000 (20%) a posledních 5.600 (30%). Výsledkem je daň 6.680.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadejte mzdu: 24000
Odpovidajici dan je: 4200
Zadejte mzdu: 35600
Odpovidajici dan je: 6680
```

Příklad 3.6

Napište program v jazyku C program, který vygeneruje 1000 náhodných hodnot v intervalu od -1000 do 1000 a zjistěte kolik je z toho:

- rovno 0
- kladných
- záporných
- v intervalu 1-100
- v intervalu 101-1000

Vypište zjištěné hodnoty a kontrolní součet.

Vzorové řešení příkladu naleznete v kapitole 15.

Příklad 3.7

Napište program **hádání čísla** v jazyku C. Program vygeneruje jedno náhodné číslo (do 20) a uživatel se snaží toto číslo uhodnout tak, že vždy zadá nějakou hodnotu a program odpoví buď:

1. zadej větší číslo, nebo
2. zadej menší číslo,

dokud uživatel neuhodne správnou hodnotu. Následně je na obrazovku vypsán počet pokusů.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadejte cislo: 50
Zadejte mensi cislo: 25
Zadejte vetsi cislo: 37
Zadejte vetsi cislo: 42

Cislo nalezeno na 4 pokusy
```

Příklad 3.8

Napište program v jazyku C, který vypíše čísla v binárním tvaru. Na vstupu je číslo v dekadickém tvaru a následně je toto číslo vypsáno v binární podobě (11 = 00001011).

Vzorové řešení příkladu naleznete v kapitole 15.

Je využito maskování nejnižšího bitu a rotace čísla.

```
          00001011
        & 00000001
výsledek  00000001  vypíšu hodnotu čísla ->1
```

Provedu rotaci: **`cislo >>1`**

```
          00000101
        & 00000001
výsledek  00000001  vypíšu hodnotu čísla ->1
```

Provedu rotaci: **`cislo >>2`**

```
          00000010
        & 00000001
výsledek  00000000  vypíšu hodnotu čísla ->0
```

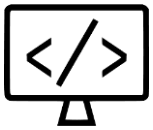
Provedu rotaci: **`cislo >>3`**

```
          00000001
        & 00000001
výsledek  00000000  vypíšu hodnotu čísla ->1
```

Atd.

4. Ukazatel void, funkce a jejich volání, pole a funkce

Tento blok se věnuje praktickému procvičení problematiky ukazatelů na void a různým úrovním ukazatelů. Dále je pozornost věnována definici a deklaraci funkcí. Jsou procvičeny různé druhy funkcí a způsoby jejich volání. Závěr bloku je věnován ukazatelům na funkce.



Příklad 4.1

Napište program v jazyku C. Mějme `int a, b, *p_a, *p_b`. Uložte do ukazatelů adresy statických proměnných *a, b, c* a dále pracujte již **pouze** pomocí ukazatelů.

- Načtěte do proměnných hodnoty (přes ukazatele).
- Vypište hodnotu proměnné (přes ukazatel), hodnotu ukazatele (tedy adresu statické proměnné, kterou obsahuje), adresu ukazatele (adresa proměnné), adresu statické proměnné.
- Sečtěte a +b (přes ukazatele).

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadejte stranu a: 8
Zadejte stranu b: 5
Promenna *p_a ukazuje na hodnotu 7, obsahuje
adresu 0x0039fd60. Adresa staticke promenne
0x0039fd64
Soucet a + b pomoci ukazatelu: 13
```

Příklad 4.2

Napište program v jazyku C, který definuje funkci, jež vrací součet posloupnosti čísel od 1 do N, (1+2+3-+…+N). Hlavní program načte hodnotu N, výsledek funkce zobrazí na obrazovce.

```
Soucet 5 cisel je: 15
```

Příklad 4.3

Napište program v jazyku C, který definuje funkci, která, spočítá kořeny kvadratické rovnice. Funkce vrací 0 v případě, že rovnice nemá reálné řešení a 1 v případě, že reálné řešení má. Hlavní program načte hodnoty a,b,c. Pokud rovnice má reálné řešení, pak se výsledky vytisknou na obrazovku. V opačném případě je na obrazovku vytištěno, že reálné řešení nemá.

Nápověda: funkce odmocniny `sqrt()` je dostupná z hlavičkového souboru `math.h`.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadej a: 1
Zadej b: 2
Zadej c: -8
Rovnice má dva koreny. x1=2, x2=-4
```

Příklad 4.4

Napište program v jazyku C, který definuje funkci, jež vrací četnost požadovaného znaku v řetězci. Hlavní program načte řetězec a hledaný znak. Výsledek je vytištěn na obrazovku.

Modifikace: funkce vrací `void`, hodnotu četnosti obsahuje proměnná předávaná pomocí adresy.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadej retezec: Ahoj jak se dnes mas?
Zadej hledany znak: s
Počet znaku s: 3
```

Příklad 4.5

Napište program v jazyku C, který definuje 3 funkce, které budou sloužit ke sčítání, odčítání a násobení dvou vstupních parametrů. Na začátku programu se uživateli zobrazí menu (pomocí funkce) s možnostmi:

- 1 – sčítání hodnot
- 2 – odečítání hodnot
- 3 – násobení hodnot

Uživatel si zvolí jednu z možností, která načte námi vytvořenou funkci do ukazatele na funkci, který pak voláme v cyklu do té doby, než uživatel zadá hodnotu 0. Program bude vypisovat jednotlivé operace na obrazovku.

Vzorové řešení příkladu naleznete v kapitole 15.

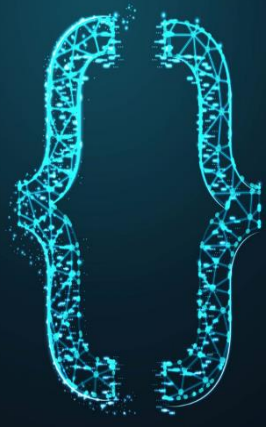
```
Zadej a:3
Zadej b:5
1 – scitani hodnot
2 – odecitáni hodnot
3 – nasobeni hodnot
Vyber operaci: 1
Vysledek vybrane operace: 8
```

Příklad 4.6

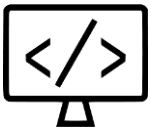
Napište program v jazyku C, který definuje funkci, jež provede konverzi desítkového čísla do binární podoby.

```
Zadej dekadicke cislo: 8
Cislo v bin. hodnote: 1111
```

5. Konstanty, ukazatele a jednorozměrné statické pole



Tento blok se věnuje praktickému procvičení problematiky konstant, dále se pak věnuje důležité problematice ukazatelů. Zvládnutí a porozumění ukazatelům je klíčové pro práci s poli, dynamickou pamětí, soubory a mnoha dalšími. Poslední část bloku se věnuje praktickému procvičení jednorozměrného statického pole.



Příklad 5.1

Napište program v jazyku C. Mějme `int a, b, *p_a, *p_b`. Uložte do ukazatelů adresy statických proměnných a dále pracujte již **pouze** pomocí ukazatelů.

- Načtěte do proměnných hodnoty (přes ukazatele).
- Vypište hodnotu proměnné (přes ukazatel), hodnotu ukazatele (tedy adresu statické proměnné, kterou obsahuje), adresu ukazatele (adresa proměnné), adresu statické proměnné.
- Sečtěte a +b (přes ukazatele).

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadejte stranu a: 8
Zadejte stranu b: 5
Promenna *p_a ukazuje na hodnotu 7, osahuje
adresu 0x0039fd60. Adresa staticke promenne
0x0039fd64
Soucet a + b pomoci ukazatelu: 13
```

Příklad 5.2

Napište program v jazyku C. Zadefinujte konstantu rozmers=5 (pomocí makra), definujte statické pole o velikosti N. Načtěte prvky do pole. Program umožňuje:

- Vypsat prvky pole.
- Najít Max/Min a jejich indexy -> informace uložte to příslušných proměnných a vypište.
- Prohodit Max s Min a vypsat pole.

Vzorové řešení příkladu naleznete v kapitole 15.

Alternativně řešte pomocí funkcí.

```
Zadej 1. prvek: 1
Zadej 2. prvek: 2
Zadej 3. prvek: 3
Zadej 4. prvek: 4
Zadej 5. prvek: 5

Vypis pole: 1,2,3,4,5

Prohozene pole: 5,2,3,4,1
```

Příklad 5.3

Napište program v jazyku C program, který definuje funkci, jež, spočítá kořeny kvadratické rovnice. Funkce vrací 0 v případě, že rovnice nemá reálné řešení a 1 v případě, že reálné řešení má. Hlavní program načte hodnoty a,b,c. Pokud rovnice má reálné řešení, pak se výsledky vytisknou na obrazovku. V opačném případě je na obrazovku vytištěno, že reálné řešení nemá. Hodnoty x1, x2 řešte pomocí pole o dimenzi dva prvky.

Nápověda: funkce odmocniny `sqrt()` je dostupná z hlavičkového souboru `math.h`.

```
Zadej a: 1
Zadej b: 2
Zadej C: -8
Rovnice má dva koreny. x1=2, x2=-4
```

Příklad 5.4

Napište program v jazyku C, který definuje funkci se vstupním parametrem pole celých čísel a počtem prvků. Tato funkce vrací průměrnou hodnotu prvků, které obsahuje. Pole celých čísel načtěte z klávesnice. Velikost pole je dáno konstantou. Výsledek zaokrouhlete na dvě desetinná místa.

```
Zadej 1. prvek: 1
Zadej 2. prvek: 2
Zadej 3. prvek: 3
Zadej 4. prvek: 4
Zadej 5. prvek: 5
Zadej 6. prvek: 6
Zadej 7. prvek: 7
Zadej 8. prvek: 8
Zadej 9. prvek: 8

Prumer pole je: 4.89
```

Příklad 5.5

Napište program v jazyku C, který definuje funkci, jež provede reverzi vstupního pole hodnot. Vstupními parametry funkce je pole vstupních hodnot a jeho velikost. Prvky pole načtěte z klávesnice a vypište před a po zavolání vámi definované funkce. Velikost pole je dána konstantou.

Tip: Řešte pomocí pomocného lokálního pole.

```
Zadej 1. prvek: 1
Zadej 2. prvek: 2
Zadej 3. prvek: 3
Zadej 4. prvek: 4
Zadej 5. prvek: 5
Zadej 6. prvek: 6

Vypis pole: 1,2,3,4,5,6

Vypis pole po reverzi:6,5,4,3,2,1
```

Příklad 5.6

Napište program v jazyku C, který definuje funkci, jež provede kopii pole. Vstupními parametry jsou:

1. Vzorové pole (obsahuje prvky)
2. Cílové pole (prázdné pole)
3. Dimenze

Funkce provede zkopírování prvků do nového pole. Funkce vrací 1 v případě úspěchu, hodnotu 0 pak v případě neúspěšné operace. Hodnoty vstupního pole jsou zadány z klávesnice. Po volání funkce je nově naplněné pole vypsáno.

```
Zadej 1. prvek: 1
Zadej 2. prvek: 2
Zadej 3. prvek: 3
Zadej 4. prvek: 4
Zadej 5. prvek: 5
Zadej 6. prvek: 6

Vypis pole noveho pole B:1,2,3,4,5,6
```

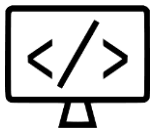
Příklad 5.7

Napište program v jazyku C, který zjistí velikost statického pole. Využijte operátor sizeof.

```
Velikost pole je: 6 prvku
```

6. Dynamické alokace, realokace, vícerozměrná pole

Tento blok se věnuje praktickému procvičení problematiky dynamické alokace, realokace a uvolňování paměti. Dále je věnována pozornost alokaci paměti ve vztahu k jednorozměrným ale i ke dvourozměrným polím.



Příklad 6.1

Napište program v jazyku C, který dynamicky alokuje jednorozměrné pole celých N čísel. Pole načtěte a následně vypište na obrazovku. Dále proveďte realokaci pole na dvojnásobnou dimenzi, doplňte pole dalšími hodnotami a opět vypište na obrazovku.

Vzorové řešení příkladu naleznete v kapitole 15.


```
Zadej 1. prvek: 1
Zadej 2. prvek: 2
Zadej 3. prvek: 3
Zadej 4. prvek: 4
Zadej 5. prvek: 5

Vypis pole: 1,2,3,4,5

Zadej 6. prvek: 6
Zadej 7. prvek: 7
Zadej 8. prvek: 8
Zadej 9. prvek: 9
Zadej 10. prvek: 10


Vypis realokovaneho pole: 1,2,3,4,5,6,7,8,9,10
```

Příklad 6.2

Napište program v jazyku C, který definuje sadu funkcí pro práci s plně dynamickým dvourozměrným polem - 2D (tedy ukazatel na pole ukazatelů, které ukazují na int) Požadované funkce:

Alokuj	provede alokaci dvourozměrného pole (řádky, sloupce) a vrací ukazatel na alokovanou paměť
Nacti	načte 2D pole celými čísly
Dealokuj	dealokace celého 2D pole (obrácený postup alokace)
Vypis	vytiskne hodnoty 2D pole
Najdi	hledá zadanou hodnotu v 2D poli, vrací 0 v případě neúspěchu a 1 v případě, že se hledaný prvek v poli vyskytuje
Prohod	prohodí maximální prvek s minimálním
MaxSloupec	zjistí, který sloupec má největší součet prvků
Nahrad Sud	nahradí sudé prvky hodnotou 0

Hlavní program načte sloupce a řádky a provede postupně:

Alokuj – Nacti – Vypis – Zadej hledanou hodnotu: Najdi (o výsledku je uživatel informován) –Prohod – NahradSud - MaxSloupec – Dealokuj

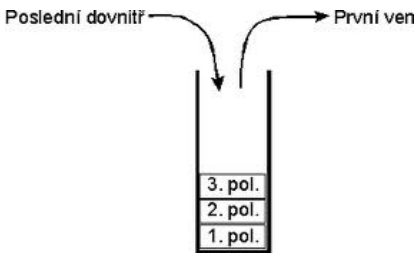
Vzorové řešení příkladu naleznete v kapitole 15.

Příklad 6.3

Napište program v jazyku C, který odráží datovou strukturu zásobník. Tedy prvek, který byl jako poslední vložený, je na vrcholu zásobníku a při následné operaci odeber je právě tento prvek ze zásobníku odebrán. Jsou požadovány funkce:

Vlož	vloží prvek do zásobníku
Odeber	odebere prvek ze zásobníku
Vypiš	vypíše obsah zásobníku

Datová struktura zásobník je realizovaná pomocí dynamického pole a jednotlivé prvky jsou celá čísla.



```
Vloz:1
Vloz:2
Vloz:3
Vloz:4
Vloz:5


Vypis:
5
4
3
2
1

Odeber

Vypis:
4
3
2
1
```

Příklad 6.4

Napište program v jazyku C, který požadovaný finanční obnos rozdělí dle dostupných platidel (5000, 2000, 1000, 500, 200, 100, 50, 20, 10, 5, 2, 1). Tedy na příklad částku 11235 rozloží na 5000, 5000, 1000, 200, 20, 10, 5. Tento problém řešte pomocí funkce, která jako vstupní parametr „obdrží“ danou částku. Výstupním parametrem funkce je dynamické pole, které obsahuje jednotlivá platidla. Hlavní program následně toto pole vypíše na obrazovku.

```
Zadej castku: 11235
Pouzita platidla:  5000,5000,1000,200,20,10,5
```

Příklad 6.5

Napište program v jazyku C, který definuje funkci pro součin matic. Vstupem funkce jsou rozměry maticeA, rozměry maticeB a samotné maticeA a B. Výstupem funkce je nová dynamicky alokovaná matice obsahující součin matic A a B.

Prototyp funkce může mít následující podobu : `int **soucin(int mA, int nA, int **matA, int mB, int nB, int **matB)`

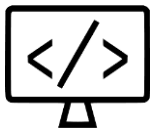
```
Matice A:
1 2 3
4 5 6

Matice B:
1 0
2 1
0 -1

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

7. Práce se znaky a řetězci

Tento blok se věnuje praktickému procvičení problematiky znaků a vstupně výstupním funkcím, které se znaky pracují. Dále je pozornost věnována poli znaků, jakožto představiteli řetězců, vstupně výstupním funkcím a funkcím z hlavičkového souboru `string.h`, které umožňují s řetězci pohodlněji pracovat.



Příklad 7.1

Napište program v jazyku C, který bude čekat na vstupu z klávesnice slovo od uživatele. Následně program bude vypisovat jednotlivé znaky řetězce na obrazovku.

V modifikaci řešení rozdělte do dvou funkcí:

- Načti řetězec
  - Výstupní parametr- dynamicky alokované pole znaků
- Vypiš řetězec po znacích
  - Vstupní parametr-pole znaků

```
Zadej slovo: telefon
Vypis po znacích: t e l e f o n
```

Příklad 7.2

Napište program v jazyku C, který provede rozbor věty zadané z klávesnice. Program bude provádět statistiku počtu jednotlivých samohlásek (a, e, i, o, u, y). Musí být ošetřeno jak zadání malého písmena, tak i velkého písmena (je tedy zpracováno jak malé, tak velké písmeno). Na závěr program vypíše statistiku samohlásek.

K ukládání statistiky využijte pole, které bude obsahovat prvky pro statistiku jednotlivých samohlásek.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadej vetu: Napis program v jazyce C.
Rozbor:
a:3
e:1
i:1
o:1
u:0
y:1
```

Příklad 7.3

Napište program v jazyku C, který načte libovolný řetězec o konstantní délce (včetně mezer). Zjistěte délku řetězce pomocí funkce z knihovny `<string.h>`. V načteném řetězci nahraďte každý třetí znak nějakým zvoleným znakem (např. '_' nebo '*' atd.) a takto upravený řetězec vytiskněte na obrazovku.

```
Zadej retezec: qweasdq weasd rwe xvxd
Po uprave: qw_as_q _ea_d _we_xv_d
```

Příklad 7.4

Napište program v jazyku C pro práci s textem. Mějme následující definici:

```
char str1[20],str2[20],str3[20],zn;
char *ukStr=NULL;
int porovnej=0;
```

Načtěte str1 a str2. Zjistěte a vypište délku obou řetězců. Zkopírujte obsah str2 do str3. Dále připojte řetězec str3 k str1 a vypište. Načtěte z klávesnice znak (do proměnné zn). Zjistěte, zda se zadaný znak v řetězci str3 vyskytuje, pokud ano, tak vypište, na jaké pozici (pomocí rozdílu dvou ukazatelů `(ukStr a str3)`). Porovnejte str1 a str2 a vypište, který z nich je větší popřípadě, že jsou si rovny (využijte k rozhodnutí příkaz `switch` a proměnnou `porovnej`, která obsahuje výsledek porovnání).

Pozn.: pro práci s řetězci využijte funkce z knihovny `<string.h>`.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadej str1: Ahoj jak se dnes
Zadej str2: mas.
Delka str1 je: 16
Delka str2 je: 4
Obsah retezce str3 je: mas.
Obsah spojeni str1 a str3 je: Ahoj jak se dnes
mas.
Zadej hledany znak: s
Hledany znak se vyskytuje na pozici: 10
Porovnani retezcu: str1 < str2
```

Příklad 7.5

Napište v jazyku C funkci, která zjišťuje počet výskytů požadovaného znaku v řetězci.

Možný prototyp funkce: `int pocVyskytu(char *kde, char co)`

```
Zadej retezec: Jak se dnes mas?
Hledany znak: e
Pocet vyskytu: 2
```

Příklad 7.6

Napište v jazyku C funkci, která zjišťuje počet výskytů požadovaného podřetězce v řetězci.

Možný prototyp funkce: `int pocVyskytu(char *kde, char *co)`

```
Zadej retezec: Jak se dnes mas? Jak asi
Hledany podretezec: Jak
Pocet vyskytu: 2
```

Příklad 7.7

Napište v jazyku C funkci, která nahradí požadovaný znak zvoleným znakem. Možný prototyp funkce: `int pocVyskytu(char *kde, char co, char cim)`

```
Zadej vetu retezec: Jak se dnes mas?
Hledany znak: e
Nahradit: *

Vypis po zmene: Jak s* dn*s mas?
```

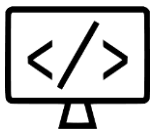
Příklad 7.8

Napište v jazyku C funkci, která definuje jeden vstupní parametr typu řetězec, který obsahuje jednotlivé položky oddělené středníkem. Tato funkce provede parsování z jednoho řádku a jednotlivé údaje uloží do pole řetězců.

```
Zadej vstupni data:
Ing;Jan;Novak;Ph.D.;vedouci divize;466046111;
Vypis polozek z pole:
Ing
Jan
Novak
Ph.D.
vedouci divize
466046111
```

8. Práce se soubory – textový

Tento blok se věnuje praktickému procvičení problematiky vstupů a výstupů na externí médium. Jsou procvičeny režimy textového a binárního souboru a funkce, které s těmito typy souborů dokáží pracovat. Tento blok se věnuje textovým souborům.



Příklad 8.1

Napište program v jazyku C, který:

- načtete větu z klávesnice
- uloží tuto větu do souboru data.txt
- čte ze souboru data.txt jednotlivé znaky, malá písmena (a-z) převede na velká a postupně je ukládá do soboru VELKE.txt a zároveň je tiskne na obrazovku

Vzorové řešení příkladu naleznete v kapitole 15.

```
Zadej vetu: Dnes vetsina lidi pouziva mobilni
telefon.

Nactena veta ze souboru po znacich ze souboru:
Dnes vetsina lidi pouziva mobilni telefon.
```

Příklad 8.2

Napište program v jazyku C, který pomocí definované funkce sčítá hodnoty reálných čísel na řádku a ukládá je do nového souboru. Na konci souboru se uloží celkový součet.

Prototyp funkce: `void soucty(const char *vstup, const char *vystup)`

Kde `vstup` je název vstupního souboru a `vystup` je název výstupního souboru.

Příklad vstupního souboru.

```
7.134 1.112 0.5198 2.436 0.9626 0.4995
1.27 1.324 0.9639 1.538
1.503 1.15 0.3411
0.111 2.133
4.287 8.675 1.231 0.2131 0.22
```

Příklad výstupního souboru

```
12.6639
5.0959
2.9941
2.244
14.6261

Suma:37.624
```

Příklad 8.3

Napište program v jazyku C, který pomocí definované funkce sčítá hodnoty reálných čísel na řádku a ukládá je do nového souboru. Na konci souboru se uloží celkový součet. Příklad je modifikací předchozího příkladu. V tomto případu však funkce definuje jako vstupní parametr jednotlivé řádky. Z těchto řádků získá hodnoty, které následně sečte. Funkce vrací součet příslušného řádku.

Prototyp funkce: `double soucetNaRadku(const char *vstup)`

Kde `vstup` je pole znaků.

Příklad vstupního souboru.

```
7.134 1.112 0.5198 2.436 0.9626 0.4995
1.27 1.324 0.9639 1.538
1.503 1.15 0.3411
0.111 2.133
4.287 8.675 1.231 0.2131 0.22
```

Příklad výstupního souboru.

```
12.6639
5.0959
2.9941
2.244
14.6261

Suma:37.624
```

Příklad 8.4

Napište program v jazyku C, který pomocí definované funkce nejprve načte dimenzi pole. Na základě této informace se provede alokace jednorozměrného pole typu `int`. Následně jsou jednotlivé hodnoty načteny do pole. Pole je přes návratovou hodnotu vráceno do funkce main. Dimenze pole je z funkce předána technikou „odkazem“. Dále je definována funkce pro výpis, která vypíše na obrazovku načtené pole.

Vzorové řešení příkladu naleznete v kapitole 15.

Možný prototyp funkce:

```
int *nactiPole(const char *jmSoub,int *dimPole)
```

Příklad vstupního souboru.

```
5
1 2 3 4 5
```

Výstup na obrazovku.

```
Nactene pole:
1
2
3
4
5
```

Příklad 8.5

Napište následující funkce v jazyku C program:

- Alokuj – alokuje a naplní náhodnými čísly 2D matici. Tato funkce vrací matici přes návratový typ. Rozměry matice jsou předány pomocí ukazatele na proměnou.
- UložX2 – která uloží do souboru rozměry matice a samotnou matici (vstupní parametr), jejichž jednotlivé hodnoty prvků jsou dvojnásobkem hodnot původních.

Příklad matice .

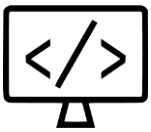
```
5 3 5 2
6 2 9 3
3 7 8 6
```

Příklad výstupu do souboru..

```
3 4
10 6 10 4
12 4 18 6
6 14 16 12
```

9. Práce se soubory – binární

Tento blok se věnuje praktickému procvičení problematiky vstupů a výstupů na externí médium. Jsou procvičeny funkce, které dokáží detekovat konce binárních souborů.



Příklad 9.1

Napište program v jazyku C, který:

- dynamicky alokuje a naplní pole1 náhodnými hodnotami
- uloží do binárního souboru data.bin velikost pole a následně celé pole (pole jako celý blok najednou)
- dále otevře soubor data.bin, načte velikost pole a na základě této hodnoty alokuje pole2
- ze souboru načte hodnoty do pole2

Využijte funkce `fread, fwrite`

Vzorové řešení příkladu naleznete v kapitole 15.

Příklad 9.2

Napište program v jazyku C, který nejprve zapíše do binárního souboru náhodný počet celých čísel (například 10-50 záznamů).

Dále definujte funkci, která bez znalosti počtu záznamů zjistí počet celých čísel v souboru. Na základě této informace alokuje jednorozměrné pole, které je následně naplněno hodnotami ze souboru. Pole následně vypište na obrazovku.

Tip: k zjištění počtu čísel využijte znalost, že velikost jednoho celého čísla je obvykle 4B. Dále přesun v souboru a funkci, která udává v B aktuální pozici.

Vzorové řešení příkladu naleznete v kapitole 15.

```
Nactene pole:
3,44,33,213,54,23,234,332,121,1,3
```

Příklad 9.3

Napište program v jazyku C, který využívá připravený vstupní soubor předchozího příkladu.

Definujte funkci, která ze souboru přečte náhodně čtyři hodnoty, které uloží do pole. Načtené hodnoty vypíše na obrazovku a pole setřídí. Setříděné pole opět vypíše na obrazovku

Vzorové řešení příkladu naleznete v kapitole 15.

Ilustrativní vstupní binární souboru.

```
3,44,33,213,54,23,234,332,121,1,3
```

Výstup na obrazovku.

```
Nahodne nactene hodnoty:
54
44
234
121

Vypis setrideneho pole:
44 54 121 234
```

Příklad 9.4

Napište program v jazyku C program, který následující funkce:

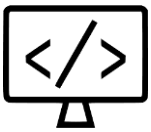
NactiMatici — vstupní parametry jsou rozměry matice. Hodnoty jsou náhodně generovány v intervalu (0-100). Alokovaná a načtená matice tvoří návratovou hodnotu funkce.

UlozMatici — vstupní parametr je matice a její rozměry. Funkce uloží rozměry matice a matici do souboru prvek po prvku.

NactiMatici — vstupní parametr tvoří název souboru. Funkce načte rozměry matice a alokuje prázdnou matici. Dále funkce načte ze souboru hodnoty **řádek po řádku**. Načtená matice tvoří návratovou hodnotu funkce.

10. Struktury a unie

Tento blok se věnuje praktickému procvičení problematiky zapouzdření heterogenních záznamů do datového typu struktura. Jsou procvičeny možné definice struktur, struktury v jiné struktuře, je nastíněna problematika struktury a funkce.

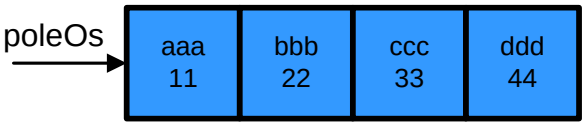


Příklad 10.1

Napište program v jazyku C, který umožnuje pracovat s polem osob. Nadefinujte nový datový typ struktury OSOBA a konstantu definující počet osob.

```
typedef struct osoba{
     char jmeno[30];
     int vek;
}tOsoba;
```

Vytvořte pole osob, dále načtěte jednotlivé osoby z klávesnice. Vyhledejte a vypište nejstarší osobu.




```
Zadej 1. osobu:
Jmeno:aaa
Vek:11
Zadej 2. osobu:
Jmeno:bbb
Vek:22
Zadej 3. osobu:
Jmeno:ccc
Vek:33
Zadej 4. osobu:
Jmeno:ddd
Vek:44

Nejstarsi osoba je: ddd,44
```

Příklad 10.2

Napište program v jazyku C, který umožnuje pracovat se zřetězenými záznamy. Nadefinujte nový datový typ struktury tOsoba, jenž umožňuje odkazovat na sama sebe.

Vzorové řešení příkladu naleznete v kapitole 15.

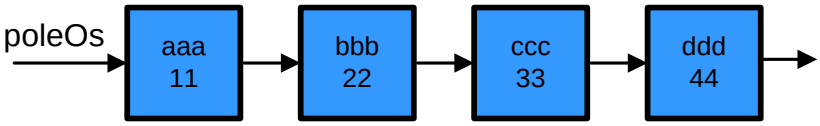
```
typedef struct osoba{
     char jmeno[30];
     int vek;
     struct osoba *dalsi;
}tOsoba;
```

Definujte následující funkce:

Alokuj	která vrací ukazatel na alokovanou strukturu.
Načti	jež načte jednotlivé položky struktury (předání adresy struc.).
Vypiš	která vypíše všechny záznamy.
Dealokuj	která provede dealokaci všech záznamů.
Najdi	která vyhledá požadovanou osobu.

Pomocí těchto funkcí alokujte a naplňte alespoň tři osoby.

Poz.: Jednotlivé osoby je nutno spojovat přes položku další




```
Zadej 1. osobu:
Jmeno:aaa
Vek:11
Zadej 2. osobu:
Jmeno:bbb
Vek:22
Zadej 3. osobu:
Jmeno:ccc
Vek:33
Zadej 4. osobu:
Jmeno:ddd
Vek:44

Hledej osobu: xxx
Osoba xxx neni v seznamu osob


Hledej osobu: ccc
Nalezena osoba: ccc,33
```

Příklad 10.3

Napište program v jazyku C, který umožnuje pracovat se zřetězenými záznamy reprezentující nějaké produkty. Nadefinujte nový datový typ struktury tProdukt, jež umožňuje odkazovat na sama sebe.

Vzorové řešení příkladu naleznete v kapitole 15.

```
typedef struct produkt{
     char jmenoProduktu[30];
     int cena;
     int mnozstvi;
     struct produkt *dalsi;
}tProdukt;
```

Připravte si bázový soubor s daty, viz následující vzor.

```
Produkt1
Produkt2
Produkt3
Produkt4
Produkt5
Produkt6
Produkt7
Produkt8
```

Dále definujte následující funkce:

NactiProdukty	která nejprve zjistí počet položek a následně alokuje pole produktů. Dále postupně čte názvy produktů a ukládá je do pole produktů. Cena a množství je **náhodně** generováno (cena <10,500> množství <0,200>).
DoplnStavZasob	která doplní stav zásob produktům, kterých je méně než 20.
Odeber	která odebere požadované množství (vstupní parametr) náhodného produktu.
VypisProdukty	která provede výpis všech produktů, jejich ceny a množství.

Hlavní program (funkce main) obsahuje načtené pole záznamů a volá výše uvedené funkce v pořadí:

- nactiPordukty
- vypisProdukty
- doplnStavZasob
- odeber
- vypisProdukty


```
jmProduktu:Produkt1
cena:123
mnozstvi: 100
jmProduktu:Produkt2
cena:223
mnozstvi: 399
jmProduktu:Produkt3
cena:231
mnozstvi: 190
jmProduktu:Produkt4
cena:532
mnozstvi: 232
jmProduktu:Produkt5
cena: 981
mnozstvi: 211
jmProduktu:Produkt6
cena:748
mnozstvi: 18
jmProduktu:Produkt7
cena:771
mnozstvi: 112
jmProduktu:Produkt8
cena:212
mnozstvi: 88
```

Příklad 10.4

Rozšiřte předchozí (Produkty o adresu) příklad o **vnořený** záznam tAdresa, viz vzor.


```
typedef struct{
 char ulice[20];
 int cp;
}tAdresa;
```

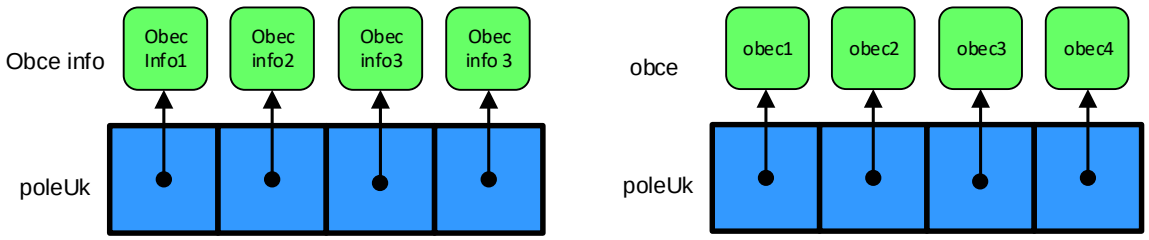
Příklad 10.5

Napište program v jazyku C, postavený na **poli ukazatelů na záznam**, který z binárního souboru *obce.bin* **náhodně načte** osm obcí (**přímo ze souboru**). Dále pro tyto obce ze souboru *obceInfo.bin* načtěte doplňující informace o každé obci. Záznamy v *obceInfo.bin* jsou seřazeny podle ID.

Vzorové řešení příkladu naleznete v kapitole 15.

```
typedef struct obec{
     char nazevObce[20];
     int id;
}tObec;

typedef struct obecInfo{
     int pocetOb;
     int cisloKraje;
     int id;
}tObecInfo;
```



Definujte jednotlivé funkce:

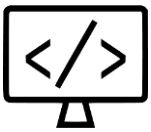
nactiObce	která alokuje pole ukazatelů pro 8 obcí. Dále jsou náhodně ze souboru *obce.bin* načteny jednotlivé obce. Alokované a naplněné pole tvoří návratovou hodnotu funkce.
Setrid	která setřídí obce dle ID.
nactiObceInfo	která alokuje pole ukazatelů. Vstupní parametr tvoří pole obcí. Na základě informace ID obce, načtěte ze souboru *obceInfo.bin* doplňující informace o každé obci. (ID obce odráží pořadí záznamu v soboru *obceInfo.bin*). Alokované a naplněné pole je návratovou hodnotou funkce.
Vypis	která vypíše všechny obce a její doplňující informace.
Dealokuj	která dealokuje všechny dynamické alokace.

funkce `main` volá pouze funkce:

- nactiObce
- setrid
- nactiObceInfo
- vypis
- dealokuj

11. Preprocesor a makra

Tento blok se věnuje praktickému procvičení činnosti preprocesoru, jeho základní použití. Dále jsou procvičeny standardní makra a makra uživatelská jak s parametry, tak bez parametrů.



Příklad 11.1

Napište program v jazyku C, který pomocí standardního makra zjistí, kdy byl zdrojový kód přeložen a jaký je vstupní soubor.

```
Tento program byl prelozen 12.1.2012
Vstupni soubor je pokus.c
```

Příklad 11.2

Napište program v jazyku C, který pomocí makra s parametry definuje sadu „inline“ funkcí základních matematických operací.

- Součet
- Rozdíl
- Součin
- Dělení
- Zbytek po dělení

Program dále obsahuje uživatelské menu, které načte operandy a následně uživatel zvolí typ požadované operace.

```
1-Soucet
2-Rozdíl
3-Soucin
4-Podíl
5-Zbytek po deleni
Zvolte operand A:15
Zvolte operand B:2
Typ operace: 4

Vysledek operace: 7.5
```

Příklad 11.3

Napište program v jazyku C, který pomocí předefinovaných maker pro práci s písmem a konverzi písmene čte postupně znaky z textového souboru, které testuje, zda odpovídají velkému písmenu. Pokud odpovídají malému písmenu, pak je pomocí makra převede na velké písmeno. Všechny znaky jsou postupně vypsány na obrazovku.

Příklad bázového textového souboru.

```
První veta. Druha o neco delsi veta. Treti je
jeste o neco delsi nez druha.
```

Výstup na obrazovku

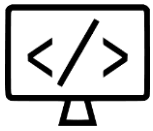
```
PRVNÍ VETA. DRUHA O NECO DELSI VETA. TRETI JE
JESTE O NECO DELSI NEZ DRUHA.
```

Příklad 11.4

Napište program v jazyku C, který využije podmíněného překladu, pro potřeby ladění aplikace. V tomto případě režimu `DEBUG` se definuje pole o velikosti pěti čísel, které si definujeme. Pokud není aplikace v režimu `DEBUG`, pak se volá definovaná funkce `Nacti`, která načítá prvky do pole pomocí funkce náhodných čísel.

12. Paměťové třídy, oddělený překlad větších projektů

Tento blok se věnuje praktickému procvičení platnosti identifikátorů a popisu jednotlivých paměťových tříd. Dále je procvičen oddělený překlad, který je základním kamenem všech větších projektů a jenž umožňuje udržet větší přehlednost.



Příklad 12.1

Napište program v jazyku C, který definuje funkci, jež počítá počet volání. Po každém zavolání funkce vypište, po kolikáté byla funkce zavolána.

```
Fce volana: 1x
Fce volana: 2x
Fce volana: 3x
Fce volana: 4x
```

Příklad 12.2

Napište program v jazyku C Telefonní seznam pracující nad jednosměrně zřetězeným seznamem. Prvky seznamu jsou jednotlivé osoby. Připravte si textový soubor seznam.csv. Soubor má následující organizaci:

```
Id;titulPred;jméno;příjmení;titulZa;pobocka;tel;mail;
```

Příklad textového souboru.

```
1;Ing.;Vaclava;Mala;Ph.D.;centrala;571674683;aubrechtova@aaa.cz
2;;Eva;Babicova;;pobocka;571674417;babicova@aaa.cz
3;Mgr.;Katerina;Bajcikova;;pobocka;571674195;bajcikova@aaa.cz
4;;Roman;Bambuch;;pobocka;571274521;bambuch@aaa.cz
```

Modul osoba:
Tento modul definuje strukturu tOsoba.

osoba.h

```
typedef struct osoba{
     char jmeno[20];
     char prijmeni[20];
     char titulPred[7];
     char titulZa[7];
     char telefon[20];
     char mail[30];
     struct osoba *dalsi;
}tOsoba;
```

a deklaraci funkcí

- `vytvorOsobu` Vstupní parametry tvoří jednotlivé údaje. Alokovaná a naplněná struktura je rovněž návratová hodnota
- `vypisOsobu`, vypíše všechny informace o osobě

osoba.c – definuje příslušné funkce

Hlavní modul s funkcí main
Tento modul obsahuje definici funkcí a hlavní funkci main

- `nacti` která postupně ze souboru (vstupní parametr) čte jednotlivé řádky, ze kterých získává požadované informace. Využívá funkci `vytvorOsobu` a funkci `vloz`.
- `vloz` která vloží novou osobu do seznamu na první pozici do lineárního seznamu, nová osoba je zde jako vstupní parametr.
- `vypis` která:
  - vypíše všechny osoby v seznamu, využívá funkci `vypisOsobu`.
  - vypíše osoby obsahující zadaný řetězec (vstupní parametr).

Hlavní modul s funkcí main
telSeznam.c – Sekvenčně volá funkce

- `nacti` načte osoby ze souboru a vkládá je do seznamu
- `vypis` vypíše osoby ze seznamu
- `vypis` vypíše osoby osahující daný řetězec (ve jméně)

Globální proměnné nejsou přípustné

Příklad 12.3

Napište program v jazyku C ZPRÁVY, kde jsou položky tvořeny jednotlivými záznamy/zprávami. Položky jsou lineárně řetězeny a vyšší priorita má vyšší číslo. Struktura projektu bude obsahovat následující moduly:

`zprava.h` – obsahuje

- definici strukturu `tZprava` s datovými položkami
  - `ID`
  - `priorita`
  - `nazev`
  - `dalsi`

    a

- deklaraci funkcí
  - `vytvorZpravu` s parametry `ID,priorita` a `nazev`, která vrací alokovanou a naplněnou strukturu.
  - `vypisZpravu`, jenž vypisuje zprávu, na kterou dostane ukazatel.

`zprava.c` – definuje příslušné funkce

`seznamZprav.h` – obsahuje

- deklaraci funkcí
  - `nacti`, která načítá data ze souboru, alokuje zprávu a postupně je pomocí funkce `vloz` vkládá do seznamu.
  - `vypis`, která vypíše všechny zprávy v seznamu.
  - `odeber`, která odebere jednu zprávu s nejvyšší prioritou, tuto zprávu vrací do volající funkce main.
  - `najdi`, která vrací do volající fce main vyhledanou zprávu dle názvu.
  - `zrus`, která dealokuje všechny zprávy s seznamu.
  - `uloz`, která uloží všechny zprávy v seznamu do souboru.

`seznamZprav.c` – uchovává ukazatel na první zprávu a dále definuje příslušné funkce a definuje funkci:

- `vloz`, která vloží novou zprávu do seznamu na první pozici, nová zpráva je zde jako vstupní parametr

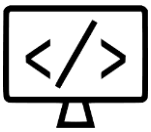
`main.c` – obsahuje menu s možnostmi:

- `nacti` načte všechny zprávy ze souboru do seznamu.
- `vypiš` vypíše všechny zprávy.
- `odeber` odebere zprávu, vypíše ji a následně dealokuje.
- `najdi` vyhledá zprávu dle názvu.
- `zruš` dealokuje všechny zprávy v seznamu.
- `ulož` uloží zprávy do souboru.

Vzorové řešení příkladu naleznete v kapitole 15.

13. Standardní knihovny a funkce

Tento blok se věnuje praktickému procvičení vybraných funkcí z některých hlavičkových souborů.



Příklad 13.1

Napište v jazyku C program KALKULACKA využijte knihovnu `math.h`. Program umožnuje provádět následující sadu operací:

- Aritmetické operace
  - Součet
  - Rozdíl
  - Součin
  - Dělení
  - Zbytek po dělení
  - Absolutní hodnotu
  - n-tou
    - mocninu
    - odmocninu

- Goniometrické operace
  - Sin
  - ArcSin
  - Cos
  - ArcCos
  - Tang
  - ArcTang

Uživatel zadá pomocí uživatelského menu požadovanou operaci a následně operand/y.

```
1-Aritmeticke operace
2-Goniometricke operace

Volba:1

1-Součet
2-Rozdíl
3-Součin
4-Dělení
5-Zbytek po dělení
6-Absolutní hodnota
7- n-mocninu
8- n-odmocninu
Volba:1
Zadejte A:7
Zadejte B:9

Vysledek operace: 16
```

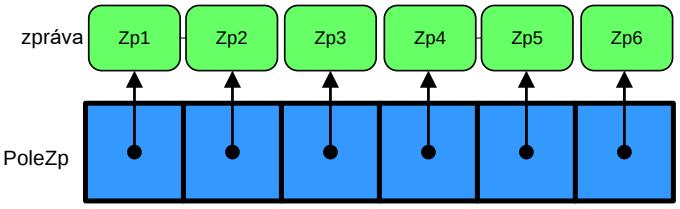
Příklad 13.2

Vytvořte program v jazyce C ZPRAVY, který je realizovaný pomocí pole ukazatelů na záznam, kde jsou položky tvořeny jednotlivými záznamy/prioritními zprávami. Položky jsou uloženy v poli ukazatelů na zprávu a vyšší prioritu má vyšší časové razítko. Struktura projektu bude obsahovat následující moduly:

`zprava.h` – obsahuje

```
typedef struct zprava{
     int ID;
     struct tm cas;
     char teloZpravy[100];

}tZprava;
```



- deklaraci funkcí
  - `vytvorZpravu` s parametry `ID`, `cas` a `nazev`, která vrací alokovanou a naplněnou strukturu.
  - `vypisZpravu`, jenž vypisuje zprávu, na kterou dostane ukazatel.

`zprava.c` – definuje příslušné funkce

`seznamZprav.h` – obsahuje

- deklaraci funkcí
  - `nacti`, která nejdříve zjistí počet záznamů, dále alokuje pole ukazatelů a následně alokuje a načítá data z binárního souboru jednotlivé záznamy.
  - `vypis`, která vypíše všechny zprávy z pole.
  - `odeber`, která odebere jednu zprávu s nejvyšší prioritou, tuto zprávu vrací do funkce main, prázdné místo (ukazatel v poli) je nastaveno na hodnotu NULL.
  - `vlož`, která vloží zprávu na první volné místo, pokud není volné místo, provede se realokace pole na dvojnásobnou velikost.
  - `najdi`, která vrací do fce main vyhledanou zprávu dle názvu.
  - `zrus`, která dealokuje všechny zprávy.

`seznamZprav.c` – uchovává pole ukazatelů na zprávy, počet záznamů, velikost pole a dále definuje příslušné funkce :

`main.c` – obsahuje menu s možnostmi:

- nacti – načte všechny zprávy ze souboru do seznamu
- vypiš – vypíše všechny zprávy
- vloz – vloží novou zprávu
- odeber – odebere zprávu, vypíše ji a následně dealokuje
- najdi – vyhledá zprávu dle názvu
- zruš – dealokuje všechny zprávy ve frontě

Příklad 13.3

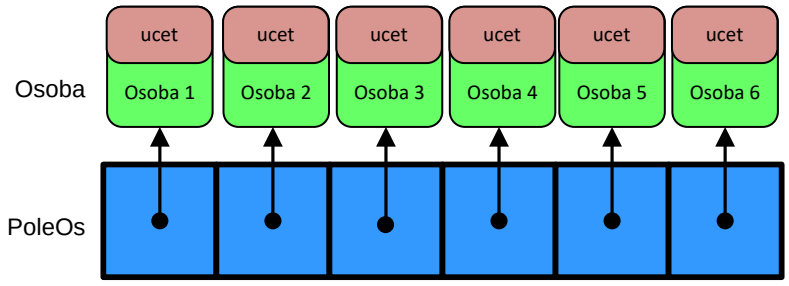
Vytvořte v jazyce C program ***Banka*** využívající pole osob vlastnících účet.

Vzorové řešení příkladu naleznete v kapitole 15.

Modul účet:

Tento modul definuje strukturu účet:

```
ucet.h
typedef struct ucet{
     int id;
     int zustatek;
     struct tm cas;
 }tUcet;
```



Modul osoby:
Tento modul definuje strukturu osoba a deklaruje funkce:

poleOsob.h

```
typedef struct osoba{
     char jm[20];
     char pr[20];
     tUcet ucet;
 }tOsoba;

NactiPoleOS();
Setrid(int typ);
VypisOsoby();
```

poleOsob.c
Udržuje dynamické **pole ukazatelů** na osoby.

- `NactiPoleOS` – načte ze souboru *osoby.txt* seznam osob. Jednotlivé osoby postupně alokuje a ukládá do pole ukazatelů na osobu.
- `Setrid` – setřídí pole, typ parametru určuje, zda osoby budou setříděny dle příjmení, zůstatku na účtu, či podle poslední operace (času)
- `Vypis` – vypíše pole osob

Hlavní modul s funkcí main
Sekvenčně volá funkce

- `NactiPoleOS();`
- `Setrid(typ);`
- `VypisOsoby();`

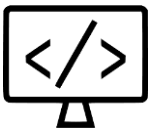
V tomto modulu nejsou již definovány žádné další funkce.

Vzorová struktura bázového souboru: jméno, číslo účtu, zůstatek, datum poslední operace (time).

```
Jana
Babicova
10017
213000
1373803428
Katerina
Bajcikova
10011
739000
1373802128
```

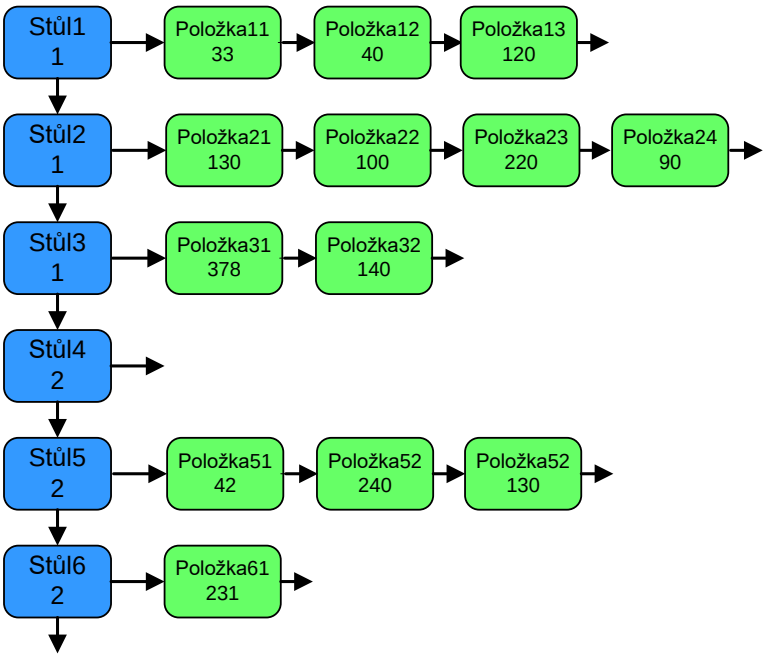
14. Komplexnější příklady

Tento blok se věnuje praktickému procvičení komplexnějších částí jazyka C. Příklady jsou dobrým procvičením na zápočet a zkoušku.



Příklad 14.1

Vytvořte aplikaci ***restaurace*** postavenou nad dynamicky alokovanými a lineárně zřetězenými záznamy. V aplikaci existují dva druhy záznamů, (i) **stoly** (s datovými položkami ***název stolu*** a ***číslo číšníka***) a (ii) **položky** (s datovými položkami ***název*** položky a ***cena*** položky). Jednotlivé stoly jsou obsluhovány číšníky. Každá položka se vztahuje vždy k nějakému stolu. Aplikace se bude skládat z několika jednotlivých modulů a hlavní funkce `main` budu disponovat sadou operací pro práci se stoly a jednotlivými položkami. Základní koncepce aplikace je uvedena na následujícím obrázku.



Modul stůl:

Tento modul umožňuje základní správu stolů, které jsou v seznamu stolů a spolupracuje s modulem `polozka.h`.

`stul.h`


```
typedef struct stul{
      char nazevStolu[20];
      int CisloCisnik;
      struct stul *dalsiStul;
      struct polozka *prvPolozka;
}tStul;

void VytvorStul(char *nazev,int cisnik);
void OdeberAktStul();
void OdeberPolozkyAktStolu();
void ZmenCisnika(int NovCisnik);
int NajdiStul(char *nazev);
void VypisStoly();
void VypisAktStul();
void VlozPolozku(char *nazev,int cena);
void VypisPolNaAktStole();
int CelkovaUtrataNaAktStole();
int PocetPolozekNaAktStole();
int CisnikSnejvicPol();
void VypisVse();
```

stul.c

```
typedef struct polozka{
      char polzkaJmeno[20];
      int cena;
      struct polozka *dalsi;
}tPolozka;


tPolozka *VytvorPolozku(char *nazev,int cena);
void VypisPolozku(tPolozka *pol);
```

Uchovává ukazatele na začátek seznamu stolů, na aktuální stůl a definuje jednotlivé funkce deklarované v modulu `stul.h`.

```
tStul* prvni=NULL; tStul* akt=NULL;
```

- `VytvorStul` – vloží alokovanou a naplněnou strukturu na poslední místo v seznamu stolů. Vstupními parametry jsou název stolu a číslo číšníka, který stůl obsluhuje. Název musí být vždy jedinečný (např. Stů1).
- `OdeberAktStul` – odebere aktuální stůl ze seznamu stolů, pokud jsou na stole nějaké položky je nejdříve volána funkce OdeberPolozkyAktStolu
- `OdeberPolozkyAktStolu` – z aktuálního stolu postupně odebere všechny položky (provádí postupnou dealokaci položek)
- `ZmenCisnika` – změní u aktuálního stolu číslo obsluhujícího číšníka.
- `NajdiStul` – vyhledá stůl dle vstupního parametru. Vrací 0 v případě neúspěchu hledání, v opačném případě vrací 1 a nastaví vyhledaný stůl jako aktuální.
- `VypisStoly` – vypíše na obrazovku všechny stoly
- `VypisAktStul` – vypíše informace o aktuálním stolu
- `VlozPolozku` – vloží nově alokovanou položku (pomocí funkce `VytvorPolozku`) na první místo v seznamu položek aktuálního stolu. Vstupními parametry jsou název a cena položky.
- `VypisPolNaAktStole` – vypíše všechny položky aktuálního stolu
- `CelkovaUtrataNaAktStole` – vrací celkovou útratu na aktuálním stole
- `PocetPolozekNaAktStole` – vrací celkový počet položek na aktuálním stole
- `CisnikSnejvicPol` – vrací číslo číšníka, který má na svých stolech nejvíce položek, přičemž využívá funkci `PocetPolozekNaAktStole`
- `VypisVse` – vypíše všechny stoly včetně všech jejich položek.

Modul položka:

Tento modul definuje strukturu položka a funkce potřebné pro jejich základní správu.

`polozka.h`

`polozka.c`

Definuje jednotlivé funkce deklarované v modulu `polozka.h` a to následovně:

- `VytvorPolozku` – alokuje a vrací novou položku, vstupními parametry jsou jméno položky a její cena.
- `VypisPolozku` – vypíše položku, jež do funkce vstupuje jako vstupní parametr.

Hlavní modul s funkcí main

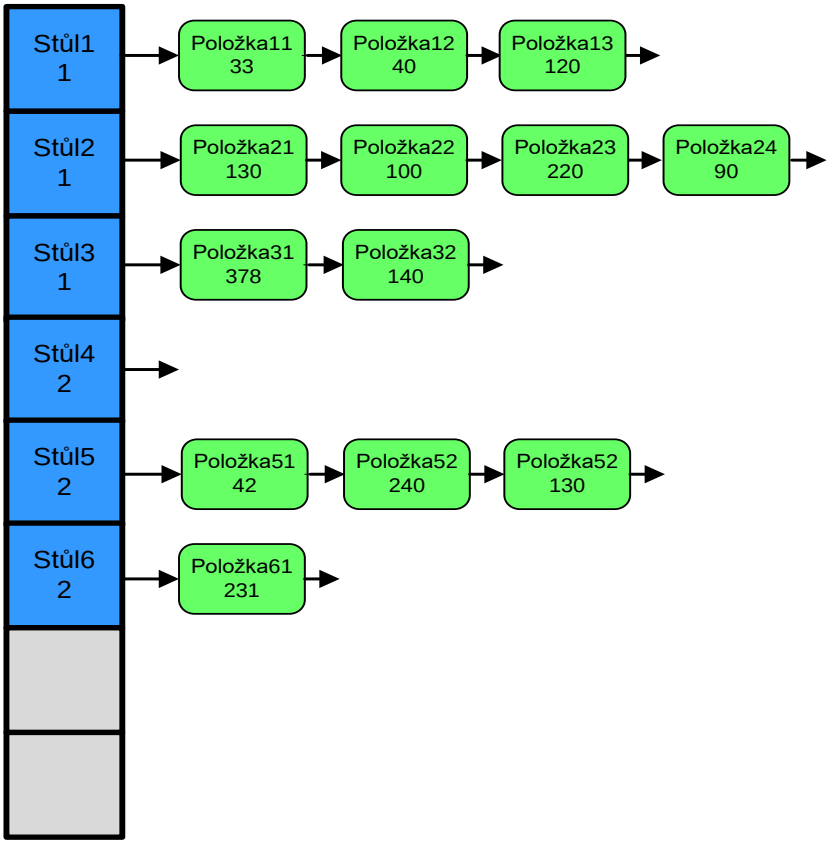
Osahuje textové menu pro obsluhu programu a volá z modulu stoly příslušné funkce

- **Vlož stůl** – založí nový stůl .
- **Odeber akt. stůl** – odebere aktuální stůl včetně všech položek.
- **Vypiš stoly** – vypíše všechny stoly.
- **Najdi stůl** – vyhledá požadovaný stůl.
- **Vlož položku na akt. stůl**– vloží novou položku do seznamu položek aktuálního stolu.
- **Výpis položek na akt. stole** – vypíše všechny položky na aktuálním stole.
- **Vypiš celkovou cenu všech položek na akt. stole** – cena všech pol. na akt. stole.
- **Vypiš číšníka s nejvíce položkami** – vypíše číslo číšníka, který má v součtu nejvíce položek na všech svých stolech.
- **Vypiš vše** – vypíše všechny stoly včetně všech jejich položek.

V tomto modulu nejsou již definovány žádné další funkce.

Příklad 14.2

proveďte modifikaci dle následujícího obrázku.

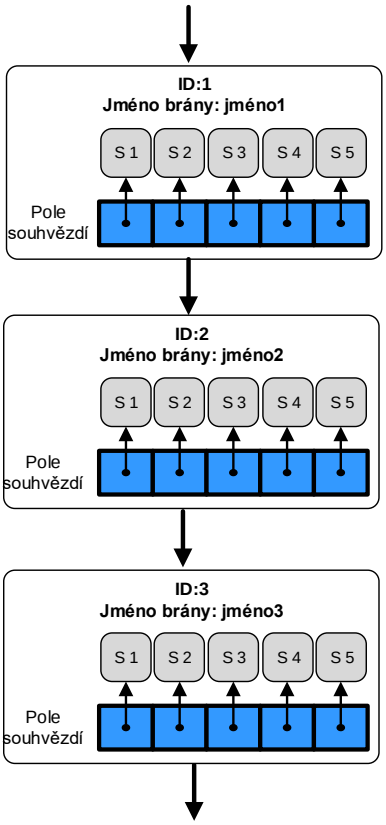


Příklad 14.3

Vzorové řešení příkladu naleznete v kapitole 15.

Star Gate – DHD

Implementujte jednoduché DHD, které umí zadáním sedmi symbolů (resp. názvů souhvězdí) vytvořit stabilní červí díru do jiného světa. Vzhledem k tomu, že nevlastníme původní antické DHD, ale vytváříme vlastní, bude jeho implementace obsahovat databázi světů pro jednodušší zadávání (a protože seržant Harriman je sklerotik a nezná adresy nazpaměť). Po úspěšném navázání spojení nás průzkumný tým informuje o stavu světa. Jednotlivé světy jsou reprezentovány pomocí lineárně zřetězeného seznamu záznamů/struktur, kde každý záznam obsahuje pole souhvězdí (řetězce).



Struktura programu:

Modul Svět

Svet.h – definice struktury `svet` a deklarace funkcí

```
typedef struct svet {
      char nazevSveta[30];
      int cisloSveta;
      char *zprava;
      char **poleSouhvezdi;//pole retezcu
      struct svet *dalsi;
} tSvet;
```

- `vytvorSvet` – alokuje strukturu a veškeré řetězce. Návratovou hodnotu funkce tvoří **ukazatel** na alokovanou strukturu
- `zrusSvet` – dealokuje dynamické řetězce a celou strukturu, vstupní parametr zastupuje **ukazatel** na strukturu
- `vytvorKopii` – vrací hlubokou kopii struktury, tedy **ukazatel** na novou strukturu se sejnými daty

- **Svet.c** – definice výše uvedených funkcí

Modul SeznamSvetu

SeznamSvetu.h – deklarace funkcí

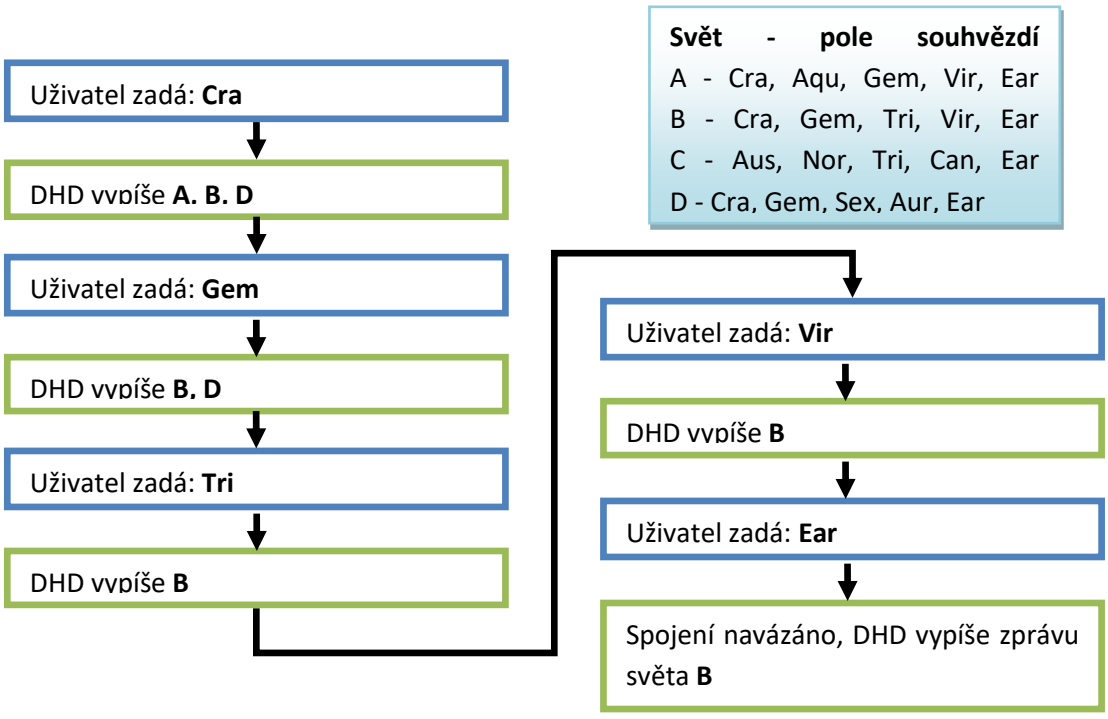
- `nactiSvety` – načte ze souboru (*data.csv)* všechny světy a vybuduje z nich jednosměrný lineárně zřetězený seznam. Vstupní parametr tvoří jméno souboru. Textový soubor má strukturu:

```
id;jmSveta;zpráva;souhv1;souhv2;souhv3;souhv4;souhv5;souhv6;souhv7
```

kde položky **`id;jmSveta;zpráva`** jsou vždy uvedeny, ale počet souhvězdí se může potenciálně **změnit**. Proto je nutné, aby se počet souhvězdí zjišťoval na základě vstupních dat. Pro správnou alokaci PoleSouhvězdí, obsahující dynamické řetězce, je nutné znát:

  - Počet souhvězdí
  - Délku každého řetězce

- Funkce `nactiSvety` využívá funkci `vytvorSvet`.
- `najdiSvet` – najde svět dle názvu. Vstupní parametr tvoří hledaný svět. Nalezený svět tvoří návratovou hodnotu funkce.
- `smazSvety` – zruší celý seznam světů, využívá funkci `zrusSvet`
- `smazHistorii` – zruší celý seznam historie, využívá funkci `zrusSvet`
- `vytvorDiru` – uživatel postupně zadává jednotlivá souhvězdí. Po každém zadaném souhvězdí se uživateli zobrazí pouze ty světy, které na aktuální pozici toto souhvězdí obsahují (a zároveň souhlasí i všechna předchozí zadaná souhvězdí). Pokud jsou všechna souhvězdí správná, uživateli se zobrazí zpráva nalezeného světa. Dále se pomocí funkce `vytvorKopii` vytvoří kopie nalezeného světa a uloží se do seznamu historie.



Příklad funkce vytvorDiru

- `vypisSvety` - vypíše všechny světy.
- `vypisHistorii` - vypíše všechny světy, které byly otevřeny pomocí správné kombinace souhvězdí.
- `vypisSvetySouh` - vypíše všechny světy, které obsahují na libovolném místě hledané souhvězdí.
- `pridejSvet` - přidá nový svět.
- `odeberSvet` - odebere svět dle názvu světa.

SeznamSvetu.c – tento soubor definuje statické proměnné pro hlavní seznam světů (`static tSvet *seznamSvetu`) a seznam hledaných světů (`static tSvet *historie`) a definuje výše uvedené funkce

Modul main

Obsahuje hlavní funkci `main` a jednoduché textové menu, které umožnuje volat funkce z modulu **SeznamSvetu**

- Načti světy ze souboru
- Vypiš světy – Hlavní seznam/historie
- Vypiš světy, které obsahují hledané souhvězdí
- Vytvoř červí díru
- Přidej nový svět
- Odeber svět dle názvu
- Najdi svět dle názvu světa
- Zruš světy – hlavní seznam/historie

15. Řešení vybraných příkladů

Tento blok poskytuje k vybraným příkladům z přechozích kapitol jejich řešení. První číslo vždy reprezentuje kapitolu, za tečkou pak následuje číslo příkladu v dané kapitole.

Příklad 3.1

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


int main(){
    int opak=1;
    int a,b,c;
    //printf("%d",time(NULL));
    srand(time(NULL));
    while(opak){
        a=rand()%10;
        b=rand()%10;
        c=rand()%10;
        if((a+b>c)&&(a+c>b)&&(c+b>a)){
            printf("Lze sestrojit s a=%d b=%d, c=%d",a,b,c);
            opak=0;
        }
        else{
        printf("NElze sestrojit s a=%d b=%d, c=%d\n",a,b,c);
        }
    }



        return 0;
}
```

Příklad 3.5

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void){
    int mzda,i;
    int h1=10000,h2=20000;
    float prD1=0.1,prD2=0.2,prD3=0.3;
    float d=0,d1=0,d2=0,d3=0;

    srand(time(NULL));
    for(i=0;i<10;i++){
        mzda=(rand()%366+85)*100; // 8500-45000
        d=0;d1=0;d2=0;d3=0;
        if(mzda<h1){
            d1=mzda*prD1;
        }
        else{
            d1=h1*prD1;
            if(mzda>h2){
                d2=h1*prD2;
                d3=(mzda-h2)*prD3;
            }
            else{
                d2=(mzda-h1)*prD2;
            }
        }

        d=d1+d2+d3;
        printf("osoba %d ma mzdu:%d a zpaltila dan:%.2f\n",i+1,mzda,d);
    }


    return 0;
}
```

Příklad 3.6

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

// generuje znaky (48-122) pocita male velke cisla a ost.znaky
int main(){

   int i;
   int n=100;
   char zn;
   int pocVel=0,pocMal=0,pocZnaku=0,pocCis=0;


   srand(time(0));
   for( i = 0; i < n; i++ ){
      zn =48+rand()%75;
      printf( "jako cislo  %3d, jako znak %3c \n",zn,zn );


       if((48<=zn)&&(zn<=57))
           pocCis++;
       else{
          if(((58<=zn)&&(zn<=64))||((91<=zn)&&(zn<=96)))
             pocZnaku++;
          else{
             if((65<=zn)&&(zn<=90))
                pocVel++;
             else{
                if((97<=zn)&&(zn<=122))
                   pocMal++;
             }
          }
       }
   }

  printf( "bylo vygenerovano:velkych: %2d malych: %2d cisel: %2d ost.znaky:
%2d",pocVel,pocMal,pocCis,pocZnaku);

   system("pause");
}
```

Příklad 3.7

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


int main(){
    int nahCislo;
    int uhodl = 0, odhad, pocetPok = 0;
    srand(time(NULL));
    nahCislo = rand() % 21;

    while (uhodl==0)
    {
        printf("zadej odhad:");
        scanf("%d", &odhad);
        if (odhad > nahCislo){
            printf("hledane je mensi\n");
            pocetPok++;
        }
        else{
            if (odhad < nahCislo){
                printf("hledane je vetsi\n");
                pocetPok++;
            }
            else{
                printf("trefa na %d.pokus\n",pocetPok);
                uhodl = 1;
            }
        }
    }


    return 0;

}
```

Příklad 3.8

```c
#include <stdio.h>


int main(){

   int a,b,c;
   int pom,i;
   int maska=1;
   printf("Zadej dekadicke cislo: ");
   scanf("%d",&a);

   printf("cislo %d je  binarne ",a);
   for(i=7;i>=0;i--){
      pom=(a>>i)&maska;
      printf("%d",pom);
      if(i==4)
         printf(" ");
   }

   printf("\nZadej dekadicke cislo: ");
   scanf("%d",&b);
   printf("cislo %d je  binarne ",b);
   for(i=7;i>=0;i--){
      pom=(b>>i)&maska;
      printf("%d",pom);
      if(i==4)
         printf(" ");
   }


   printf("\njejich binarni soucin je ");
   c=a&b;
   for(i=7;i>=0;i--){
      pom=(c>>i)&maska;
      printf("%d",pom);
      if(i==4)
         printf(" ");
   }

   printf("\njejich binarni soucet je ");
   c=a|b;
   for(i=7;i>=0;i--){
      pom=(c>>i)&maska;
      printf("%d",pom);
      if(i==4)
         printf(" ");
   }

}
```

Příklad 4.1

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define rozmer 5
int main(int argc, char* argv[]){
    int i,a,*p_a=&a,b,*p_b=&b,soucet=0;

    a=5;
    b=6;

    soucet=*p_a+*p_b;

    printf("adresa p_a:%X hodnota na jeji adrese: %d, adresa ukazatele p_a:%X, adresa a:%X\n",p_a,*p_a,&p_a,&a);
    printf("adresa p_b:%X hodnota na jeji adrese: %d, adresa ukazatele p_b:%X, adresa b:%X\n",p_b,*p_b,&p_b,&b);
    printf("adresa souctu=%X a jeho hodnota je %d\n",&soucet,soucet);
    return 0;
}
```

Příklad 4.2

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

int soucetCisel(int n){
    int i,soucet=0;
    for(i=1;i<=n;i++){
        soucet+=i;
    }
    return soucet;
}


int main(int argc, char* argv[]){
    int pocetCis,souc=0;
    printf("zadej N: ");
    scanf("%d",&pocetCis);

    souc=soucetCisel(pocetCis);
    printf("soucet %d cisel je: %d",pocetCis,soucetCisel(pocetCis));
    return 0;
}
```

Příklad 4.3

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


int KvadrRov(int a, int b, int c, double *body)
{
   double D;
   D = b*b - 4 * a*c;
   if (D >= 0){
      body[0] = (-b + sqrt(D)) / (2 * a);
      body[1] = (-b - sqrt(D)) / (2 * a);
      return 1;
   }
   else
      return 0;
}
int KvadrRov2(int a, int b, int c, float *xx1, float *xx2 )
{
   double D;
   D = b*b - 4 * a*c;
   if (D >= 0){
      *xx1 = (-b + sqrt(D)) / (2 * a);
      *xx2= (-b - sqrt(D)) / (2 * a);
      return 1;
   }
   else
      return 0;
}

void main()
{
   double BodyX[] = { 0, 0 };

   if (KvadrRov(1, 2, -8, BodyX) == 1)
      printf("Koreny jsou: x1=%.2lf x2=%.2lf", BodyX[0], BodyX[1]);
   else
      printf("Nema reseni v R");
   float x1 = 0, x2 = 0;
   if (KvadrRov2(1, 2, -8, &x1,&x2) == 1)
      printf("Koreny jsou: x1=%.2f x2=%.2f", x1,x2);
   else
      printf("Nema reseni v R");

}
```

Příklad 4.4

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

#define N 50

int CetnostZn(char string[],char znak,int *cetnostZnRef){
    int i,cetnost=0,delkaStr;
    delkaStr=strlen(string);

    for(i=0;i<delkaStr;i++){
        if(string[i]==znak)
            cetnost++;
    }
    *cetnostZnRef=cetnost;
    return cetnost;
}

int main(){
    char str[N],zn;
    int cetRef=0;

    printf("Zadej retezec: ");
    gets(str);
    printf("Zadej znak: ");
    zn=getchar();

    printf("Cetnos znaku %c, je %d",zn,CetnostZn(str,zn,&cetRef));
    return 0;
}
```

Příklad 4.5

```c
/*
   Napiste tri funkce, ktera budou slouzit ke scitani, odcitani a nasobeni
   dvou vstupnich parametru. Na zacatku programu uzivateli vyskoci menu (pomoci
   zavolani funkce pro menu) s moznostmi:
      1 - scitani hodnot
      2 - odcitani hodnot
      3 - nasobeni hodnot
   Uzivatel zvoli jednu z moznosti, ktera nacte nami vytvorenou funkci do ukazatele na funkci,
   ktery budeme volat v cyklu do te doby, nez uzivatel nezada hodnotu 0. Program bude vypisovat
   jednotlive operace na obrazovku. (napr. 1+2=3+3=6...)
*/
#include <stdio.h>

void menu(){
   printf("MENU:\n");
   printf("1 - scitani hodnot\n");
   printf("2 - odcitani hodnot\n");
   printf("3 - nasobeni hodnot hodnot\n");
   printf("\nZvolte moznost: ");
}

int soucetHodnot(int a, int b){
   int vysledek = a + b;
   printf("%d + %d = %d\n", a, b, vysledek);
   return vysledek;
}

int odecitaniHodnot(int a, int b){
   int vysledek = a - b;
   printf("%d - %d = %d\n", a, b, vysledek);
   return vysledek;
}

int nasobeniHodnot(int a, int b){
   int vysledek = a * b;
   printf("%d * %d = %d\n", a, b, vysledek);
   return vysledek;
}

void main(){
   int volba;
   int b = 1, a = 0;
   int (*pFunkce)(int, int);

   menu();
   scanf("%d", &volba);
   switch (volba)
   {
   case 1:
      pFunkce = soucetHodnot;
      break;
   case 2:
```

```
        pFunkce = odecitaniHodnot;
        break;
    case 3:
        pFunkce = nasobeniHodnot;
        break;
    }
    while(1) {
        printf("Zadejte hodnotu: ");
        if(a == 0)
            scanf("%d", &a);
        else {
            scanf("%d", &b);
            if(b == 0)
                break;
            a = pFunkce(a, b);
        }
    }
}
```

Příklad 5.1

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char* argv[]){
    int i,a,*p_a=&a,b,*p_b=&b,soucet=0;

    a=5;
    b=6;

    soucet=*p_a+*p_b;

    printf("adresa p_a:%X hodnota na jeji adrese: %d, adresa ukazatele p_a:%X, adresa a:%X\n",p_a,*p_a,&p_a,&a);
    printf("adresa p_b:%X hodnota na jeji adrese: %d, adresa ukazatele p_b:%X, adresa b:%X\n",p_b,*p_b,&p_b,&b);
    printf("adresa souctu=%X a jeho hodnota je %d\n",&soucet,soucet);
   return 0;
}
```

Příklad 5.2

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define _CRTDBG_MAP_ALLOC
#include <crtdbg.h>


#define rozmer 5
int main(int argc, char* argv[]){
    int pole[rozmer];
    int max,iMax,iMin,min,pom;


    for(int i=0;i<rozmer;i++){
        printf("Zadej %d. prvek: ", i+1);
        scanf("%d",pole+i);
        //pole[i]=rand()%100;

    }

    for(int i=0;i<rozmer;i++){
        printf("prvek[%d] = %d\n",i+1, *(pole+i));

    }
    max=pole[0];
    iMax=0;
    for(i=0;i<rozmer;i++){
        if(pole[i]>max){
            max=pole[i];
            iMax=i;
        }
    }
    printf("max je = %d a je na indexu %d\n",max,iMax+1);
```

```
    min=pole[0];
    iMin=0;
    for(i=0;i<rozmer;i++){
        if(pole[i]<min){
            min=pole[i];
            iMin=i;
        }
    }
    printf("min je = %d a je na indexu %d\n",min,iMin+1);


    pom=pole[iMax];
    pole[iMax]=pole[iMin];
    pole[iMin]=pom;

    for(int i=0;i<rozmer;i++){
        printf("prvek[%d] = %d\n",i+1, *(pole+i));

    }
    printf("velikost pole je=%d", sizeof(pole)/sizeof(pole[0]));



return 0;
}
```

Příklad 6.1

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



int main(){
    int* poleDyn;
    int i, pocet = 5;


    poleDyn = (int*)calloc(pocet, sizeof(int));

    for (i = 0; i < pocet; i++)
        poleDyn[i] = i;

    for (i = 0; i < pocet; i++)
        printf("%d ", poleDyn[i]);
    printf("\n");
    int pocet2 = 2 * pocet;

    //realoace
    realloc(poleDyn, pocet2 * sizeof(int));

    for (i = pocet; i < pocet2; i++)
        poleDyn[i] = i*10;

    for (i = 0; i < pocet2; i++)
        printf("%d ", poleDyn[i]);

    system("pause");

    return 0;
}
```

Příklad 6.2

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int **Alokuj(int rr,int ss){
    int i,**matD;

    matD=(int**)malloc(rr*sizeof(int));
    for(i=0;i<rr;i++)
        matD[i]=(int*)calloc(ss,sizeof(int));

return matD;
}

void Dealokuj(int **matD,int rr){
    int i;
    for(i=0;i<rr;i++)
        free(matD[i]);
    free(matD);

}
void Nacti(int **matD,int rr,int ss){
    int i,j;
    srand(time(0));
    for(i=0;i<rr;i++)
        for(j=0;j<ss;j++)
            matD[i][j]=10+(rand()%(100-10));
}

void Vypis(int **matD,int rr,int ss){
    int i,j;
    for(i=0;i<rr;i++){
        for(j=0;j<ss;j++){
            printf("%d ",matD[i][j]);
        }
        printf("\n");
    }

}

int Najdi(int **matD,int rr,int ss,int prv){
    int i,j;
    for(i=0;i<rr;i++){
        for(j=0;j<ss;j++){
            if(matD[i][j]==prv){
                return 1;
            }
        }
    }
    return 0;
}
```

```c
int main(){
    int ** matDyn;
    int pocet=5,prvek;
    int r=3,s=2,a;
    printf("Zadej pocet radku:");
    scanf("%d",&r);
    printf("Zadej pocet sloupcu:");
    scanf("%d",&s);
    matDyn=Alokuj(r,s);
    Nacti(matDyn,r,s);
    Vypis(matDyn,r,s);
  //dalsi funkce doplnte


    printf("Zadsej hledany prvek: ");
    scanf("%d",&prvek);

    if(Najdi(matDyn,r,s,prvek))
        printf("prvek tam je\n");
    else
        printf("Neni\n");


    Dealokuj(matDyn,r);

return 0;

}
```

Příklad 7.2

```
#include <stdio.h>
include <string.h>
#define delkaStr 50
int main(){
    char vzor[]="aeiouy";
    int vzorPocet[6] = { 0 };
    char str[delkaStr];
    gets(str);

    int delkaVzor = strlen(vzor);
    for (int i=0; i < delkaStr; i++){
        for (int j = 0; j < delkaVzor; j++){
            if (str[i] == vzor[j] || str[i] == (vzor[j] - 32))
                vzorPocet[j]++;
        }
    }
    for (int i=0; i < delkaVzor; i++){
        printf("%c:%d\n", vzor[i], vzorPocet[i]);
    }
        system("pause");

    return 0;
}
```

Příklad 7.3

```
#include <stdio.h>
#include <string.h>
#define delkaStr 50
int main(){
    char str[delkaStr];
    int i,delka=0;

    gets(str);
    delka=strlen(str);

    for(i=0;i<delka;i++){
        if(!((i+1)%3)){
            str[i]='_';
        }
    }

    puts(str);


        system("pause");
    return 0;
}
```

Příklad 7.4

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

int main(){
   char str1[20],str2[20],str3[20],zn;
   char *ukStr=NULL;
   int porovnej=0;

   printf("zadej str1: ");
   gets(str1);
   printf("zadej str2: ");
   gets(str2);

   printf("delka str1 je: %d\n",strlen(str1));
   printf("delka str2 je: %d\n",strlen(str2));

   strcpy(str3,str2);
   printf("obsah retezce str3 je:%s\n",str3);

   strcat(str1,str3);
   printf("obsah spojeni str1 a str3 je:%s\n",str1);

   printf("Zadej hledany zank: ");
   zn=getchar();
   ukStr=strchr(str1,zn);
   if(ukStr!=NULL)
      printf("je tam, a je na pozici %d",(ukStr-str1)+1);

   else
      printf("Neni");

   porovnej=strcmp(str1,str2);
   switch(porovnej){
      case 0:
         printf("\nRovno");
         break;
      case -1:
         printf("\nstr1 < str2");
         break;
      case 1:
         printf("\nstr1 > str2");
         break;
   }

      system("pause");
   return 0;
}
```

Příklad 8.1

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


int main(){
   FILE *f,*f2;
    char zn;

    char str[50];
    printf("Zadej vetu\n");
    gets(str);

    f=fopen("./data.txt","w");
    fputs(str,f);
    fclose(f);

    f=fopen("./data.txt","r");
    f2=fopen("VELKE.txt","w");
    while(zn=fgetc(f),feof(f)==0){

        if(zn>='a' && zn<='z'){// je to maly znak
            fputc(zn-32,f2);
            putchar(zn-32);

        }
        else{
            fputc(zn,f2);
            putchar(zn);

        }
    }

    fclose(f2);
    fclose(f);

        system("pause");
return 0;

}
```

Příklad 8.4

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



void printArray(double *dyArr,int count){
    int i;
    for(i=0;i<count;i++){
        printf("%.1lf ", dyArr[i]);
    }
}
double* ReadArrayFormFile(char*nameOfFile,int *count){
    FILE *f;
    int i;
    double *dynArray;
    f=fopen(nameOfFile,"r");
    fscanf(f,"%d",count);
    dynArray=(double*)calloc(*count,sizeof(double));
    for(i=0;i<*count;i++){
        fscanf(f,"%lf",&dynArray[i]);
    }
    fclose(f);
    return dynArray;
}


int main(){
    double *dynArray;
    int count;

    dynArray=ReadArrayFormFile("data.txt",&count);
    printArray(dynArray,count);


        system("pause");
    return 0;
}
```

Příklad 9.1

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


int main(){
FILE *f;

   int i,*pole,*pole2;
   int pocet=5;

   pole=(int*)calloc(pocet,sizeof(int));

   for(i=0;i<pocet;i++)
      pole[i]=i+10;

   f=fopen("./data.bin","wb");
   fwrite(&pocet,sizeof(pocet),1,f);
   fwrite(pole,sizeof(pole[1]),pocet,f);
   fclose(f);


   pocet=0;
   f=fopen("./data.bin","rb");
   fread(&pocet,sizeof(int),1,f);
   pole2=(int*)calloc(pocet,sizeof(int));
   fread(pole2,sizeof(pole[1]),pocet,f);
   fclose(f);

for(i=0;i<pocet;i++)
      printf("%d\n",pole2[i]);

      system("pause");
return 0;
}
```

Příklad 9.2 a 9.3

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>


#define N 5
void ulozCisla(int kolik){
   FILE* f;
   int i;
   f=fopen("data.bin","wb");
   for(i=0;i<kolik;i++){
      fwrite(&i,4,1,f);
   }

   fclose(f);
}

void nahodnyVyber(){

   int velSoub, pocet=0,i,nahCis,hod;
   FILE *f;
   srand(time(NULL));
   f=fopen("data.bin","rb");
   fseek(f,0,SEEK_END);
   velSoub=ftell(f);
   rewind(f);
   pocet=velSoub/4;


   for(i=0;i<3;i++){
      nahCis=rand()%pocet;
      fseek(f,nahCis* sizeof(int),SEEK_SET);
      fread(&hod,sizeof(int),f);
      printf("nahodna pozice: %d, hodnota na pozici:%d\n",nahCis,hod);
   }

   fclose(f);
}

int main()
{
   ulozCisla(66);
   nahodnyVyber();
   system("pause");


return 0;
}
```

Příklad 10.2

```
#define _CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#include <stdio.h>
#include <stdlib.h>

typedef struct osoba{
    char jmeno[30];
    int vek;
    struct osoba *dalsi;

}OSOBA;

OSOBA *Alokuj(){
    OSOBA *o;
    o=(OSOBA*)malloc(sizeof(OSOBA));
    return o;
}
void Nacti(OSOBA *o){

    printf("zadej jmeno:");
    scanf("%s",o->jmeno);
    printf("zadej vek:");
    scanf("%d",&o->vek);
    o->dalsi=NULL;

}
void Vypis(OSOBA *akt){
    while(akt!=NULL){
        printf("-----------\n");
        printf("jmeno:%s\nvek%d\n",akt->jmeno,akt->vek);
        akt=akt->dalsi;
    }
}
void Dealokuj(OSOBA *akt){
    OSOBA *pom=NULL;
    while (akt!=NULL){
        pom=akt;
        akt=akt->dalsi;
        free(pom);
    }
}

int main(int argc, char* argv[])
{
    OSOBA *os,*prvni=NULL,*akt=NULL;
    int dal=1;

    while(dal){
        os=Alokuj();
        Nacti(os);
        if(prvni==NULL)
            prvni=os;
```

```
        else
            akt->dalsi=os;
        akt=os;

        printf("Konec (0)");
        scanf("%d",&dal);
    }
    Vypis(prvni);
    Dealokuj(prvni);

    if(_CrtDumpMemoryLeaks()!=0){
        printf("Nebyla provedena dealokace");
    }

    return 0;
}
```

Příklad 10.3

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct produkt{
    char nazev[50];
    int cena;
    int mnozstvi;
}tProdukt;

tProdukt *nactiProdukty(int *pocetPolozek){
    FILE *soubor;
    tProdukt *pole;
    char dump[50];
    int i;
    struct tm cas;
    *pocetPolozek=0;
    soubor=fopen("data.txt","r");
    if(soubor==NULL){
        printf("Nepodarilo se nacist");
        return NULL;
    }
    do{
        fgets(dump,50,soubor);
        *pocetPolozek+=1;
    }while(!feof(soubor));

    printf("Nalezeno polozek %d\n",*pocetPolozek);
    fclose(soubor);
    soubor=fopen("data.txt","r");
    srand(50);
    pole=(tProdukt*)malloc(*pocetPolozek*sizeof(tProdukt));
    for(i=0; i<*pocetPolozek; i++){
        fgets(pole[i].nazev,50,soubor);
        pole[i].cena=((double)rand()/ (RAND_MAX + 1))*(490)+10;
        pole[i].mnozstvi=((double)rand()/ (RAND_MAX + 1))*(200);
    }
    fclose(soubor);
    return pole;

}
void doplnStavZbozi(tProdukt *pole,int pocetPolozek){
    int i;
    for(i=0; i<pocetPolozek; i++){
        if(pole[i].mnozstvi<20){
            pole[i].mnozstvi=200;
        }

    }
    printf("doplnovani zbozi dokonceno\n");
}
void odeber(tProdukt *pole,int mnozstvi,int pocetPolozek){
```

```
    int nah=((double)rand()/ (RAND_MAX + 1))*(pocetPolozek);
    pole[nah].mnozstvi-=mnozstvi;
    if(pole[nah].mnozstvi<0){
        pole[nah].mnozstvi=0;
    }
    printf("odebirani dokonceno\n");
}

void vypis(tProdukt *pole,int pocetPolozek){
    int i;
    for(i=0; i<pocetPolozek; i++){
        printf("Nazev produktu> %s mnozstvi: %d cena: %d\n",pole[i].nazev,pole[i].mnozstvi,pole[i].cena);

    }
}
int main(){
    tProdukt *pole;
    int pocetPolozek;
    pole=nactiProdukty(&pocetPolozek);
    vypis(pole,pocetPolozek);
    doplnStavZbozi(pole,pocetPolozek);
    odeber(pole,50,pocetPolozek);
    vypis(pole,pocetPolozek);
    free(pole);
    return 0;

}
```

Příklad 10.5

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
typedef struct obec{
    char nazevObce[20];
    int id;
}tObec;

typedef struct obecInfo{
    int pocetOb;
    int cilsoKraje;
    int id;
}tObecInfo;

void nacti(){
    FILE *f,*f2;
    char *uk;
    int i;

    tObecInfo *oi =(tObecInfo*)calloc(28,sizeof(tObecInfo));
    tObec *o =(tObec*)calloc(28,sizeof(tObec));
    f=fopen("obce.txt","r");

    for(i=0;i<28;i++){
        fscanf(f,"%d",&o[i].id);
        fgets(o[i].nazevObce,20,f);
        uk=strchr(o[i].nazevObce,'\n');
        if(uk!=NULL)
            o[i].nazevObce[uk-o[i].nazevObce]=0;

        oi[i].cilsoKraje=rand()%12+1;
        oi[i].pocetOb=rand()%100000+5000;
        oi[i].id=i+1;
    }

    for(i=0;i<28;i++){
        printf("%s %d\n",o[i].nazevObce,o[i].id);
        printf("%d %d %d\n",oi[i].id,oi[i].cilsoKraje,oi[i].pocetOb);

    }
    fclose(f);

            f=fopen("obce.bin","wb");
        f2=fopen("obceInfo.bin","wb");

                fwrite(o,sizeof(tObec),28,f);
                fwrite(oi,sizeof(tObecInfo),28,f2);
    fclose(f);
    fclose(f2);
```

```
}
void nacti2(){
    FILE *f;
    int c;
    int i;
    tObec *o =(tObec*)calloc(3,sizeof(tObec));
    tObecInfo *oi=(tObecInfo*)calloc(1,sizeof(tObecInfo));

    f=fopen("obce.bin","rb");
    c=rand()%29;
    for(i=0;i<3;i++){
        c=rand()%29;
        fseek(f,c*sizeof(tObec),SEEK_SET);
        fread(o+i,sizeof(tObec),1,f);

    }
    fclose(f);

    for(i=0;i<3;i++){
        printf("%s %d\n",o[i].nazevObce,o[i].id);
    //  printf("%d %d %d\n",oi[i].id,oi[i].cilsoKraje,oi[i].pocetOb);

    }
    f=fopen("obceInfo.bin","rb");
    for(i=0;i<3;i++){
        fseek(f,(o[i].id-1)*sizeof(tObecInfo),SEEK_SET);
        fread(oi,sizeof(tObecInfo),1,f);
        printf("%d %d %d\n",oi->id,oi->cilsoKraje,oi->pocetOb);
    }
    fclose(f);

}


int main(void){
    srand(time(NULL));
    //nacti();
    nacti2();
    return 0;
}
```

Příklad 12.3

zprava.h

```
#pragma once

typedef struct zprava{
    int ID;
    int priorita;
    char teloZpravy[100];
    struct zprava * dalsi;
}tZprava;
tZprava *createZprava(int id, int priorita,char *textZp);
void vypisZpravu(tZprava *zprava);
```

zprava.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "zprava.h"

tZprava *createZprava(int id, int priorita,char *textZp){
    tZprava *zprava=(tZprava*)malloc(sizeof(tZprava));
    zprava->ID=id;
    zprava->priorita=priorita;
    strcpy(zprava->teloZpravy,textZp);
    zprava->dalsi=NULL;
    return zprava;
}
void vypisZpravu(tZprava *zprava){
    printf("Zprava \n priorita:%d id:%d text:%s\n", zprava->priorita,zprava->ID,zprava->teloZpravy);
}
```

seznamZprav.h

```
#pragma once
#include "zprava.h"

int vloz(tZprava *zprava);
void vypis();
tZprava *odeber();//odebere jedu zpavu
tZprava *najdi(int id);
void zrus();
```

seznamZprav.c

```c
#include "seznamZprav.h"
#include <stdio.h>
#include <stdlib.h>

tZprava * prvni=NULL,*akt;

int vloz(tZprava *zprava){
    tZprava *pom,*pomPred;
    int dal=1;
    if(prvni==NULL){
        prvni=zprava;
    }
    else{

        pom=prvni;
        pomPred=prvni;
        if(pom->dalsi==NULL){//je tam jen jeden
            if(zprava->priorita>=pom->priorita){//dej za
                prvni->dalsi=zprava;
            }
            else{//dej pred
                prvni=zprava;
                zprava->dalsi=pom;
            }
        }
        else{//uz jich tam je vic
            while(dal && pom!=NULL){
                if(zprava->priorita>=pom->priorita){
                    pomPred=pom;
                    pom=pom->dalsi;

                }
                else{
                    dal=0;
                }
            }

            if(pomPred==prvni){
                if(zprava->priorita==prvni->priorita){
                    prvni->dalsi=zprava;
                }
                else{
                    prvni=zprava;
                }

            }
            else{
                pomPred->dalsi=zprava;
            }
            zprava->dalsi=pom;
        }
    }
```

```
    return 1;
}
void vypis(){
    tZprava *pom;
    pom=prvni;
    while(pom!=NULL){
        vypisZpravu(pom);
        pom=pom->dalsi;
    }
}
tZprava *odeber(){
    tZprava *pom;
    pom=prvni;
    prvni=prvni->dalsi;

    return pom;
}
tZprava *najdi(int id){
    tZprava * pom=prvni;

    while(pom!=NULL && pom->ID!=id){
        pom=pom->dalsi;
    }
    return pom;
}
void zrus(){
    tZprava * pom=prvni,*pom2=prvni;

    while(pom->dalsi!=NULL){
        pom=pom->dalsi;
        free(pom2);
        pom2=pom;

    }
    free(pom);
    prvni=NULL;
}
```

Main.c

```
#include <stdio.h>
#include <string.h>
#include "seznamZprav.h"


int main(int argc, char* argv[]){
   tZprava *zprava;
   int volba=0;
   char str[100];
   int id,priorita;
   FILE *f;

do{
      printf("\n1-vloz zpravu\n");
      printf("2-vypis zpravy\n");
      printf("3-odeber zpravu\n");
      printf("4-najdi\n");
      printf("5-nacti\n");
      printf("6-zrus\n");
      printf("0-konec\n");
      printf("\nVolba: ");
      scanf("%d",&volba);

      switch(volba){

      case 1:
         _flushall();
         printf("zadej id: ");
         scanf("%d",&id);
         _flushall();
         printf("zadej prioritu: ");
         scanf("%d",&priorita);
         _flushall();
         printf("zadej zpravu: ");
         gets(str);
         zprava=createZprava(id,priorita,str);
         vloz(zprava);

         break;
      case 2:
         vypis();
         break;

      case 3:
         zprava=odeber();
         vypisZpravu(zprava);
         free(zprava);
         break;

      case 4:
         _flushall();
         printf("zadej id: ");
```

```
            scanf("%d",&id);
            zprava=najdi(id);
            vypisZpravu(zprava);
            break;
        case 5:
            f=fopen("data.txt","r");
            while(feof(f)==0){
                fscanf(f,"%d",&id);
                fscanf(f,"%d",&priorita);
                fscanf(f,"%s",str);
                //fgets(str,20,f);
                zprava=createZprava(id,priorita,str);
                vloz(zprava);
            }
            fclose(f);
            break;
        case 6:
            zrus();
            break;
        }
    }while(volba!=0);
    return 0;
}
```

Příklad 13.3

ucet.h

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

typedef struct ucet {
    int id;
    int zustatek;
}tUcet;
```

poleOsob.h

```
#pragma once
#include <stdio.h>
#include "ucet.h"

typedef struct osoba {
    char jm[20];
    char pr[20];
    tUcet ucet;
}tOsoba;

void NactiPoleOS();
void Setrid(int typ);
void VypisOsoby();
```

poleOsob.c

```
#include <stdlib.h>
#include <string.h>
#include "poleOsob.h"

tOsoba **poleOsob;
int pocetOsob;

void NactiPoleOS() {
    FILE *f;
    char vstup[20];
    int i = 0;

    printf("Banka: Nacitam osoby ze souboru...\n");
    f = fopen("osoby.txt", "r");

    fgets(vstup, 19, f);
    pocetOsob = atoi(vstup);
    poleOsob = (tOsoba**)calloc(pocetOsob, sizeof(tOsoba*));

    while(feof(f) == 0) {
        poleOsob[i] = (tOsoba*)malloc(sizeof(tOsoba));

        fgets(poleOsob[i]->jm, 19, f);
        fgets(poleOsob[i]->pr, 19, f);

        fgets(vstup, 19, f);
        poleOsob[i]->ucet.id = atoi(vstup);
```

```
        fgets(vstup, 19, f);
        poleOsob[i]->ucet.zustatek = atoi(vstup);

        i++;
    }

}

void Setrid(int typ) {
    int i, j;
    int min;
    tOsoba *pom;

    if(typ == 1) { /
        printf("\nTridim osoby podle prijmeni...\n");
        for(i=0; i<pocetOsob; i++) { // Proch�zen� pole
            min = i;
            for(j=i+1; j<pocetOsob; j++) { // Hled�n� minima
                if(strcmp(poleOsob[min]->pr, poleOsob[j]->pr) == 1) {
                    min = j;
                }
            }

            if(i != min) { // Z�m�na
                pom = poleOsob[i];
                poleOsob[i] = poleOsob[min];
                poleOsob[min] = pom;
            }
        }

    } else if(typ == 2) { // T��d�n� podle z�statku na ��tu
        printf("\nTridim osoby podle zustatku na uctu...\n");
        for(i=0; i<pocetOsob; i++) { // Proch�zen� pole
            min = i;
            for(j=i+1; j<pocetOsob; j++) { // Hled�n� minima
                if(poleOsob[j]->ucet.zustatek < poleOsob[min]->ucet.zustatek) {
                    min = j;
                }
            }

            if(i != min) { // Z�m�na
                pom = poleOsob[i];
                poleOsob[i] = poleOsob[min];
                poleOsob[min] = pom;
            }
        }
    } else { // Chybn� parametr
        return;
    }
}

void VypisOsoby() {
    int i;
```

```
    printf("\nVypis osob:\n");
    printf("===========\n");
    for(i=0; i<pocetOsob; i++) {
        printf("Jmeno: %s", poleOsob[i]->jm);
        printf("Prijmeni: %s", poleOsob[i]->pr);
        printf("ID: %d\n", poleOsob[i]->ucet.id);
        printf("Zustatek: %d\n\n", poleOsob[i]->ucet.zustatek);
    }
}
```

main.c

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

void main() {
    NactiPoleOS();
    Setrid(1); // 1 == podle prijmeni, 2 == podle zustatku na uctu
    VypisOsoby();
}
```

Příklad 14.3

Svet.h

```
#ifndef _SVET_H
#define _SVET_H
#include <crtdbg.h>

#define NAZEV_SVETA_LEN 30
#define SYMBOL_LEN 100

typedef struct svet {
    char nazevSveta[NAZEV_SVETA_LEN];
    int cisloSveta;
    char *zprava;
    char **poleSouhvezdi;//pole retezcu
    struct svet *next;
} tSvet;

tSvet* vytvorSvet(char* nazevSveta, int cisloSveta, char* zprava, int pocetSouhvezdi);
void zrusSvet(tSvet* svet, int pocetSouhvezdi);
tSvet* vytvorKopii(tSvet* svet, int pocetSouhvezdi);

#endif
```

Svet.c

```
#include "Svet.h"
#include <stdlib.h>
#include <string.h>

tSvet* vytvorSvet(char* nazevSveta, int cisloSveta, char* zprava, int pocetSouhvezdi)
{
    tSvet* svet = malloc(sizeof(tSvet));

    strcpy_s(svet->nazevSveta, sizeof svet->nazevSveta, nazevSveta);
    svet->cisloSveta = cisloSveta;

    if (zprava)
        svet->zprava = _strdup(zprava);
    else
        svet->zprava = NULL;

    svet->poleSouhvezdi = calloc(pocetSouhvezdi, sizeof(char*));
    svet->next = NULL;

    return svet;
}

void zrusSvet(tSvet* svet, int pocetSouhvezdi)
{
    if (svet->zprava)
        free(svet->zprava);

    if (svet->poleSouhvezdi)
    {
```

```
        for (int i = 0; i < pocetSouhvezdi; i++)
            if (svet->poleSouhvezdi[i])
                free(svet->poleSouhvezdi[i]);

        free(svet->poleSouhvezdi);
    }

    free(svet);
}

tSvet* vytvorKopii(tSvet* svet, int pocetSouhvezdi)
{
    tSvet* s = malloc(sizeof(tSvet));

    strcpy_s(s->nazevSveta, sizeof svet->nazevSveta, svet->nazevSveta);
    s->cisloSveta = svet->cisloSveta;

    if (svet->zprava)
        s->zprava = _strdup(svet->zprava);
    else
        s->zprava = NULL;

    s->poleSouhvezdi = calloc(pocetSouhvezdi, sizeof(char*));
    for (int i = 0; i < pocetSouhvezdi; i++)
        if (svet->poleSouhvezdi[i])
            s->poleSouhvezdi[i] = _strdup(svet->poleSouhvezdi[i]);

    s->next = NULL;

    return s;
}
```

SeznamSvetu.h

```
#ifndef _SEZNAMSVETU_H
#define _SEZNAMSVETU_H
#include <crtdbg.h>
#include "Svet.h"
#define LINE_MAX_LEN 300

void nactiSvety(char* soubor);
void smazSvety();
void smazHistorii();
tSvet* najdiSvet(const char* nazev);
void vypisSvety();
void vypisHistorii();
void vypisSvetySouh(const char* souhvezdi);
void vytvorDiru();
void pridejSvet();
void odeberSvet();

#endif
```

SeznamSvetu.c

```c
#include "SeznamSvetu.h"
#include "Svet.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <windows.h>

static tSvet* seznamSvetu = NULL;
static tSvet* historie = NULL;
static int pocetSouhvezdi = 0;

static void pridejNaKonec(tSvet* svet, tSvet** seznam) {
    if (!*seznam)
    {
        *seznam = svet;
        return;
    }

    tSvet* tmp = *seznam;
    while (tmp->next)
        tmp = tmp->next;

    tmp->next = svet;
}

static int dejPocetSouhvezdi(FILE* f)
{
    char radek[LINE_MAX_LEN];
    fgets(radek, sizeof radek, f);

    int vyskytu = 0;
    char* uk = radek - 1;
    do {
        uk = strchr(uk + 1, ';');
        vyskytu++;
    } while (uk != NULL);

    fseek(f, 0, SEEK_SET);
    return vyskytu - 3;
}

void nactiSvety(char* soubor)
{
    FILE* f;
    if (fopen_s(&f, soubor, "r"))
        return;

    pocetSouhvezdi = dejPocetSouhvezdi(f);
    char radek[LINE_MAX_LEN];

    do {
```

```
        // id; jmSveta; zpr?va; souhv1; souhv2; souhv3; souhv4; souhv5; souhv6; souhv7
        fgets(radek, sizeof radek, f);

        int id;
        char nazevSveta[NAZEV_SVETA_LEN];
        char* context = NULL;

        char* idTok = strtok_s(radek, ";", &context);
        id = atoi(idTok);

        char* nazevSvetaTok = strtok_s(NULL, ";", &context);
        strcpy_s(nazevSveta, sizeof nazevSveta, nazevSvetaTok);

        char* zpravaTok = strtok_s(NULL, ";", &context);

        tSvet* svet = vytvorSvet(nazevSveta, id, zpravaTok, pocetSouhvezdi);
        for (int i = 0; i < pocetSouhvezdi; i++)
        {
            char* tok = strtok_s(NULL, ";", &context);
            if (tok[strlen(tok) - 1] == '\n')
                tok[strlen(tok) - 1] = 0;

            svet->poleSouhvezdi[i] = _strdup(tok);
        }

        pridejNaKonec(svet, &seznamSvetu);
    } while (!feof(f));
}

void smazSvety()
{
    tSvet* tmp;
    while (seznamSvetu)
    {
        tmp = seznamSvetu;
        seznamSvetu = seznamSvetu->next;

        zrusSvet(tmp, pocetSouhvezdi);
    }

    seznamSvetu = NULL;
}

void smazHistorii()
{
    tSvet* tmp;
    while (historie)
    {
        tmp = historie;
        historie = historie->next;

        zrusSvet(tmp, pocetSouhvezdi);
    }
```

```
    historie = NULL;
}

tSvet* najdiSvet(const char* nazev)
{
    tSvet* tmp = seznamSvetu;
    while (tmp)
    {
        if (!strcmp(tmp->nazevSveta, nazev))
            return tmp;

        tmp = tmp->next;
    }

    return NULL;
}


static void vypisSeznam(tSvet* seznam)
{
    HANDLE  hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

    tSvet* tmp = seznam;
    while (tmp)
    {
        printf("%2d. %s\n    ", tmp->cisloSveta, tmp->nazevSveta);

        FlushConsoleInputBuffer(hConsole);
        SetConsoleTextAttribute(hConsole, 8/* + 16*16 */);

        printf("( ");
        for (int i = 0; i < pocetSouhvezdi; i++)
            printf("%s%s", tmp->poleSouhvezdi[i], (i + 1 != pocetSouhvezdi ? ", " : " )\n"));

        SetConsoleTextAttribute(hConsole, 7);

        tmp = tmp->next;
    }
}

void vypisSvety()
{
    vypisSeznam(seznamSvetu);
}

void vypisHistorii()
{
    vypisSeznam(historie);
}

void vypisSvetySouh(const char* souhvezdi)
{
```

```
    tSvet* tmp = seznamSvetu;
    while (tmp)
    {
        bool obsahuje = false;
        for (int i = 0; i < pocetSouhvezdi; i++)
        {
            if (tmp->poleSouhvezdi[i] && !strcmp(tmp->poleSouhvezdi[i], souhvezdi))
            {
                obsahuje = true;
                break;
            }
        }

        if (obsahuje)
        {
            printf("%d. %s\n  ", tmp->cisloSveta, tmp->nazevSveta);
            for (int i = 0; i < pocetSouhvezdi; i++)
                printf("%s%s", tmp->poleSouhvezdi[i], (i + 1 != pocetSouhvezdi ? ", " : "\n"));
        }

        tmp = tmp->next;
    }
}

tSvet* vypisVhodne(char** symboly, int poradiSymbolu)
{
    HANDLE  hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    FlushConsoleInputBuffer(hConsole);
    SetConsoleTextAttribute(hConsole, 8/* + 16*16 */);

    tSvet* tmp = seznamSvetu;
    while (tmp)
    {
        bool vyhovuje = true;
        for (int i = 0; i <= poradiSymbolu; i++)
        {
            if (strcmp(tmp->poleSouhvezdi[i], symboly[i]))
            {
                vyhovuje = false;
                break;
            }
        }

        if (vyhovuje)
        {
            if (poradiSymbolu == pocetSouhvezdi - 1)
            {
                SetConsoleTextAttribute(hConsole, 7);
                return tmp;
            }
            else
            {
```

```
            printf("  %s (%s)\n", tmp->nazevSveta, tmp->poleSouhvezdi[poradiSymbolu + 1]);
        }
    }

    tmp = tmp->next;
  }

  SetConsoleTextAttribute(hConsole, 7);
  return NULL;
}

void vytvorDiru()
{
  fflush(stdin);
  printf("Dialing...\n");
  char** symboly = calloc(pocetSouhvezdi, sizeof(char*));
  tSvet* cil = NULL;
  for (int i = 0; i < pocetSouhvezdi; i++)
  {
    printf("Zadej %d. symbol: ", i + 1);
    symboly[i] = calloc(SYMBOL_LEN, sizeof(char));
    gets_s(symboly[i], SYMBOL_LEN);

    cil = vypisVhodne(symboly, i);
  }

  if (cil)
  {
    printf("Sedm symbolu zadano...\nBrana otevrena na planetu %s!\n", cil->nazevSveta);
    if (cil->zprava)
    {
      printf("...\n");
      printf("%s\n", cil->zprava);
    }

    tSvet* kopie = vytvorKopii(cil, pocetSouhvezdi);
    pridejNaKonec(kopie, &historie);
  }
  else
  {
    printf("Sedm symbolu zadano...\nBranu nelze otevrit.\nAdresa neni spravna!\n");
  }

  for (int i = 0; i < pocetSouhvezdi; i++)
    free(symboly[i]);
  free(symboly);
}

void pridejSvet()
{
  int id;
  char r[NAZEV_SVETA_LEN];
  char rz[LINE_MAX_LEN];
```

```
    fflush(stdin);
    printf("Zadej nazev noveho sveta: ");
    gets_s(r, sizeof r);

    printf("Zadej id noveho sveta: ");
    scanf_s("%d", &id);

    fflush(stdin);
    printf("Zadej zpravu: ");
    gets_s(rz, sizeof rz);

    tSvet* svet = vytvorSvet(r, id, rz, pocetSouhvezdi);
    for (int i = 0; i < pocetSouhvezdi; i++)
    {
        char symbol[SYMBOL_LEN];
        printf("Zadej %d. souhvezdi: ", i + 1);
        gets_s(symbol, sizeof symbol);

        svet->poleSouhvezdi[i] = _strdup(symbol);
    }

    pridejNaKonec(svet, &seznamSvetu);
    printf("Svet vytvoren!\n");
}

void odeberSvet()
{
    char r[NAZEV_SVETA_LEN];

    fflush(stdin);
    printf("Zadej nazev odebiraneho sveta: ");
    gets_s(r, sizeof r);

    tSvet* predchozi = NULL;
    tSvet* tmp = seznamSvetu;
    while (tmp)
    {
        if (!strcmp(tmp->nazevSveta, r))
        {
            if (predchozi)
                predchozi->next = tmp->next;
            else
                seznamSvetu = tmp->next;

            printf("Svet odebran!\n");
            return;
        }

        predchozi = tmp;
        tmp = tmp->next;
    }
```

```
    printf("Svet nenalezen!\n");
}
```

Main.c

```
#include <crtdbg.h>
#include "Svet.h"
#include "SeznamSvetu.h"
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

int main()
{
    nactiSvety("data.txt");
    bool cont = true;
    while (cont)
    {
        int volba;
        //system("cls");
        printf("Menu\n");
        printf("  0 - Konec\n");
        printf("  1 - Vypis svety\n");
        printf("  2 - Vypis svety obsahujici souhvezdi\n");
        printf("  3 - Vypis historii\n");
        printf("  4 - Smaz historii\n");
        printf("  5 - Vytvor diru\n");
        printf("  6 - Pridej svet\n");
        printf("  7 - Odeber svet\n");
        //printf("  0 - Konec\n");
        printf("Volba: ");
        scanf_s("%d", &volba);

        switch (volba)
        {
        case 0:
        {
            cont = false;
            break;
        }

        case 1:
        {
            vypisSvety();
            break;
        }

        case 2:
        {
            char r[SYMBOL_LEN];
            fflush(stdin);
            printf("Zadej souhvezdi: ");
            gets_s(r, sizeof r);
```

```
            vypisSvetySouh(r);
            break;
        }

        case 3:
        {
            vypisHistorii();
            break;
        }

        case 4:
        {
            smazHistorii();
            break;
        }

        case 5:
        {
            vytvorDiru();
            break;
        }

        case 6:
        {
            pridejSvet();
            break;
        }

        case 7:
        {
            odeberSvet();
            break;
        }
        }
    }

    smazSvety();
    smazHistorii();

    _CrtDumpMemoryLeaks();

    return 0;
}
```

Název	Sbírka příkladů z jazyka C
Autor	Ing. Jan Fikejz, Ph.D.
Odpovědný redaktor	Lenka Tobišková
Zveřejnění	leden 2020
Stran	101
Vydání	první
Náklad	elektronická verze

ISBN 978-80-7560-280-0 (pdf)