

# Příklady k procvičení

## Úkol 95

Napište program, který k zadané ceně připočítá 25% daň a vypíše novou cenu.

## Úkol 96

Napište program, který z konzole přečte tři malá písmena a vypíše je jako velká v obráceném pořadí. Pro znaky 'a' 'b' 'c' vypíše 'C' 'B' 'A'.

## Úkol 97

Napište program, který vypíše maximální číslo, které je možné uložit do `unsigned int` a do `signed int`.<sup>138</sup>

<sup>138</sup> -1 jako `signed int` je maximální `unsigned int` a maximální `signed int` je polovina maximálního `unsigned int`.

## Úkol 98

Jaký bude výstup následujícího kódu? Vyzkoušejte vaši domněnku a zdůvodněte výsledky.

```
i = 5;
printf("%d\n", i == 8);
printf("%d\n", i = 8);
printf("%d\n", i == 8);
```

## Úkol 99

Vytvořte program, který pro zadaný řetězec, který obsahuje matematický výraz obsahující celá čísla a základní aritmetické operace +, -, \*, /, vypíše výsledek tohoto výrazu.

Například pro takto definovaný program

```
#include <stdio.h>

int main()
{
    char retezec[] = "10+2*3";

    /* TO DO */

    return 0;
}
```

bude výsledek roven 16.<sup>139</sup>

### Úkol 100

Napište funkci, která pro vstupní řetězec, který obsahuje binární číslo vrátí toto číslo v desítkové soustavě.

### Úkol 101

Napište funkci, která pro zadané číslo v desítkové soustavě vypíše toto číslo v binární podobě.

### Úkol 102

Napište funkci s hlavičkou `void transformace(int pole[], int delka)`, která změní prvky pole podle následujících pravidel:

- Pokud je prvek dělitelný 4 a zároveň je buď menší než 50 nebo větší než 65, vynásobte 20 jeho zbytek po dělení číslem 3.<sup>140</sup>
- Pokud je index prvku v poli dělitelný 2 ale není dělitelný 4, vynásobte prvek počtem cifer tohoto čísla.
- Pokud je číslo větší než 100 nahrad'te ho číslem s číslicemi v opačném pořadí.<sup>141</sup>
- Ostatní prvky nechte beze změny.

Co bude výsledkem pro pole s prvky {62, 60, 20, 32, 68, 842, 31, 12}? Vypište prvky jako znaky.

<sup>139</sup> Pro jednoduchost budeme předpokládat, že výraz je ve správném formátu (neobsahuje nepovolené znaky, mezi dvěma operacemi je číslo, výraz začíná a končí číslem...) a není tedy potřeba ověřovat jeho správnost. Také není potřeba ověřovat dělení nulou.

<sup>140</sup> Pro 16 je výsledek 20.

<sup>141</sup> Pro prvek 123 bude výsledkem 321.

**Úkol 103**

Napište program, který pro zadané  $n$  vypíše čísla od 1 do  $n$  s tím, že místo čísel dělitelných 3 vypíše TIK, místo čísel dělitelných 5 vypíše TAK. Pokud je číslo dělitelné jak 3 i 5 vypíše TIKTAK.

**Úkol 104**

Napište funkci, které zadáte počet 1 korun, 2 korun a 5 korun a hodnotu. Funkce vrátí odpověď, zda je možné ze zadaných mincí sestavit určenou hodnotu.<sup>142</sup>

<sup>142</sup> Hlavička funkce  

```
int platba(int pocet_1, int
pocet_2, int pocet_5, int
hodnota);
```

**Úkol 105**

Napište funkci, která pro zadaný řetězec vypíše všechna slova z řetězce začínající písmenem 'a'. Slova jsou oddělena mezerou.

**Úkol 106**

Upravte předchozí funkci tak, že kromě řetězce bere jako vstup i začínající znak a vypíše všechna slova z řetězce začínající zadaným znakem.

**Úkol 107**

Upravte předchozí funkci tak, že jako vstup bere dva řetězce a vypíše všechna slova z 1. řetězce začínající 2. řetězcem. Příklad pro řetězce "bazen balon bonbon trouba" a "ba" vypíše "bazen" a "balon".

**Úkol 108**

Naprogramujte funkci, která pro zadané  $n$  vrátí  $n$ -tý prvek posloupnosti, která je zadána rekurentním vztahem:  $a_1 = 14688$ ,  $a_{n+1} = \frac{1}{2}a_n + 1200$ .<sup>143</sup>

<sup>143</sup> 10. člen je roven 2424.

**Úkol 109**

Naprogramujte funkci, která jako vstup bere 2 celočíselné kladné argumenty  $m$  a  $n$  větší rovny 2 a pracuje podle následujícího pseudokódu:

1. Vypiš  $n-2$  mezer, pak řetězec "(\\o/)"
2. Opakuj  $m$  krát:

- Na nový řádek vypiš  $n$  teček, velké  $X$  a  $n$  teček.
3. Na nový řádek vypiš  $2 \cdot n + 1$  krát  $X$ .
  4. Opakuj krok 2.

**Úkol 110**

Naprogramujte funkci `void vypis(int *pole, int zacatek, int krok, int konec)`, která vypíše prvky pole od indexu `zacatek` po index `konec` s krokem `krok`. Například pro `pole = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}` a `zacatek = 0, krok = 2, konec = 9` vypíše prvky 1, 3, 5, 7, 9.

**Úkol 111**

V následujícím kódu si vytvoříme pole, do kterého budeme ukládat celá čísla. Pole se bude samo zvětšovat, aby se do něj všechna čísla vešla. Základem této struktury bude pole (`data`), kam budeme čísla ukládat. Dále si v proměnné `hlava` budeme uchovávat informaci a počtu prvků uložených v poli.

Čísla budeme ukládat od začátku (1. bude na indexu číslo 0). `hlava` bude indexem na 1. prázdné políčko (na tento index budeme přidávat další prvek, po přidání se toto číslo zvětší).

Dále je potřeba si uchovávat informaci o velikosti alokované paměti (`velikost`). Pokud je `hlava == velikost`, musíme velikost pole zvětšit pomocí funkce `realloc()`.

Základem je následující kód:

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

int *data;
int velikost;
int hlava;

void init(int);
void uvolni();
void pridej(int);

int main()
{
    int i=7;
    init(4);
    pridej(i);
    uvolni();
    return 0;
}
```

1. Doprogramujte funkci `init()`, která inicializuje pole `data` na předaný počet prvků (alokuje pro ně paměť), funkci `pridej()` pro přidání prvku do pole a funkci `uvolni()`, která na konci programu uvolní alokovanou paměť.
2. Doprogramujte funkci na vypsání prvků v poli `data`. Zavolejte tuto funkci po každém přidání prvku do pole.
3. Nahrad'te použití globálních proměnných pomocí strukturovaného typu.
4. Doprogramujte funkci na odebrání posledního prvku v poli `data`.
5. Doprogramujte funkci, která zjistí zda se číslo předané jí jako argument nachází v datové struktuře.
6. Doprogramujte funkci, která smaže prvek předaný jí jako argument z datové struktury.
7. Upravte funkci mazání prvku tak, že pokud je počet prvků ve struktuře menší než polovina jeho velikosti, zmenší paměť alokovanou pro pole `data` na polovinu.

**Úkol 112**

Napište funkci, která pro 2 zadaná čísla vrátí, zda je možné udělat jejich podíl a pokud ano, vrátí i jejich podíl.

**Úkol 113**

Napište funkci, která bere 2 argumenty (text a podretezec). Funkce v daném textovém řetězci text vyhledá první výskyt zadaného podřetězce podretezec. Funkce vrací ukazatel na první znak nalezeného podřetězce nebo konstantu NULL, pokud podřetězec podretezec nebyl nalezen.

**Úkol 114**

Pomocí dvourozměrného pole lze reprezentovat hrací pole při piškvorkách (prázdné políčko = 0, křížek = 1, kolečko = 2). Napište funkci, která prohledá toto dvourozměrné pole a vrátí nejdelší souvislou posloupnost křížků nebo koleček

1. na řádku
2. ve sloupci
3. diagonálně

Naprogramujte hru piškvorky pro dva hráče.

Dokud v poli nebude posloupnost 5 stejných znaků (využijte předchozí funkci) se budou hráči střídát a umisťovat svůj znak do pole (na střídačku budou hráči vyzváni, aby zadali 2 souřadnice, kam chtějí umístit znak).

**Úkol 115**

Vytvořme si lineární seznam obsahující celá čísla, seznam reprezentujeme následující strukturou.

```
typedef struct _node{
    int data;
    struct _node *next;
} node;
```

Každému uzlu v seznamu bude odpovídat jedna struktura node, jejíž položka data bude obsahovat dané číslo. Celý seznam si budeme v programu pamatovat jako pointer na první uzel seznamu.

Funkce pro přidání prvku do seznamu vypadá následovně.

```
node *add(node **list, int data)
{
    node *new = malloc(sizeof(node));
    new->data = data;
    new->next = *list;
    *list = new;
    return new;
}
```

Napište funkci, která:

1. vypíše všechny prvky seznamu,
2. zjistí délku seznamu,
3. přidá prvek na konec seznamu,
4. smaže prvek na začátku seznamu,
5. smaže prvek na konci seznamu,
6. pro zadané  $i$  vypíše  $i$ -tý prvek seznamu,
7. pro zadané  $i$  vypíše  $i$ -tý prvek seznamu od konce,
8. pro zadané  $i$  smaže  $i$ -tý prvek seznamu,
9. vytvoří kopii seznamu. Funkce musí fungovat tak, že pokud změníme kopii seznamu, originální seznam se nezmění.

#### Úkol 116

Napište funkci `komplexni suma(int pocet, ...)`, která vypočítá součet předaných komplexních čísel. Počet sčítaných čísel je určen pevným parametrem `pocet`, za nímž pak ve volání funkce následují hodnoty, které má funkce sčítat. Pro práci s komplexními čísly je nutné vytvořit strukturovaný datový typ `komplexni`.

#### Úkol 117

Napište program `vyraz` vyhodnocující výpočty zapsané v obrácené polské notaci a zadané z příkazové řádky, kde každý operand nebo operátor je samostatným argumentem. Například:  
`vyraz.exe 2 3 4 + *`  
 vypočítá výraz  $2 \cdot (3 + 4)$

**Úkol 118**

Napište funkci `double akumulator(double (*fce)(double, double), double cisla[], int pocet)`, která zpracuje pomocí předané funkce `fce` hodnoty z pole `cisla`, jehož velikost je dána parametrem `pocet`. Vytvořenou funkci otestujte ve funkci `main()`. Použitými akumulačními funkcemi mohou být například funkce pro součet nebo součin dvou reálných čísel, které je ovšem pro testování potřeba dodefinovat. Detaily najdete zde: <http://jazykc.inf.upol.cz/ukazatele-na-funkce/akumulator.htm>

**Úkol 119**

Definujte strukturu se třemi prvky. První prvek bude typu `float`, druhý `char` a třetí `int`. Dále definujte union se stejnými prvky. Zjistěte adresy struktury a unionu a všech jejich položek. Dále zjistěte velikost struktury a unionu.

**Úkol 120**

Naprogramujte funkci `prevod_cisla`, která provádí převod čísel mezi dvěma obecnými pozičními soustavami. Tedy vstupem je zápis čísla  $X$  v soustavě o základu  $r$  a informace o cílovém základu  $s$ . Například pro vstup `cislo = 25, zaklad = 7, cil = 3` je výsledek roven 201. Pro jednoduchost počítejte se základy od 2 do 20.

**Úkol 121**

Naprogramujte funkci, která pro zadanou množinu (pole) prvků vypíše všechny jeho permutace.<sup>144</sup>

**Úkol 122**

Naprogramujte funkci, která pro zadanou množinu (pole) prvků a  $k$  vypíše všechny jeho  $k$  prvkové variace.

**Úkol 123**

Naprogramujte funkci, která pro zadanou množinu (pole) prvků a  $k$  vypíše všechny jeho  $k$  prvkové variace s opakováním.

<sup>144</sup>Nápověda: K řešení tohoto úkolu je možné použít rekurzi. Například všechny permutace  $n$  prvkové množiny dostaneme tak, že vezmeme jeden z prvků na první místo a pak zavoláme generování všech permutací na zbývajících  $n - 1$  prvků. Toto opakujeme pro všechny prvky.





invertuje (zamění 1 za 0 a naopak)  $n$  bitů čísla  $x$  od pozice  $p$  včetně. Ostatní bity zůstanou nezměněny.

### Úkol 129

Naprogramujte funkci, která je deklarovaná následovně `clovek* nacti_csv(char *, int *)`; . První argument představuje adresu csv souboru, který má funkce načíst do pole strukturovaného datového typu `clovek`. Druhý vstupní argument představuje ukazatel na proměnnou, do které funkce uloží počet načtených prvků ze souboru.

*csv* je jednoduchý textový formát, kde každý řádek představuje jeden záznam. Jednotlivé položky v záznamu jsou odděleny ; (na konci řádku středník není). V našem případě každý záznam obsahuje 3 položky - jméno, příjmení a věk.

Je potřeba vytvořit strukturovaný datový typ `clovek`, který bude obsahovat 3 položky (`jmeno`, `prijmeni`, `vek`). Předpokládejme, že maximální délka jména i příjmení je předem známá (například 20). Vytvořte pro tato čísla konstanty.