

## ZADÁNÍ SEMESTRÁLNÍ PRÁCE

### Celočíselná kalkulačka s neomezenou přesností

#### Zadání

Naprogramujte v jazyce ANSI C přenositelnou<sup>1</sup> **konzolovou aplikaci**, která bude fungovat jako jednoduchý **interpret aritmetických výrazů zapsaných v infixové formě**. Vstupem programu bude kromě řídicích příkazů i seznam aritmetických výrazů obsahující celočíselné operandy ( $\mathbb{Z}$ ) s neomezenou velikostí, zapsané ve dvojkové, desítkové či šestnáctkové číselné soustavě. Pro kódování operandů zapsaných ve dvojkové či šestnáctkové soustavě bude použit výhradně dvojkový doplňkový kód. Výstupem je pak odpovídající seznam výsledků vyhodnocení každého výrazu.

Program bude spouštěn příkazem `calc.exe [<vstupní-soubor>]`. Symbol `<vstupní-soubor>` zastupuje nepovinný parametr – název vstupního souboru se seznamem aritmetických výrazů. Není-li symbol `<vstupní-soubor>` uveden, program bude fungovat v interaktivním módu, kdy se příkazy budou provádět přímo zadáním z konzole do interpretu.

Vámi vyvinutý program tedy bude vykonávat následující činnosti:

1. Při spuštění bez parametru bude čekat na vstup od uživatele. Zadaný výraz vyhodnotí a bude vyžadovat další vstup, dokud nebude uveden příkaz `quit`.
2. Při spuštění s parametrem načte zadaný vstupní soubor, každý výraz v něm uvedený vypíše na obrazovku, okamžitě vyhodnotí a výsledek rovněž vypíše. Po zpracování posledního výrazu dojde k ukončení programu s návratovou hodnotou `EXIT_SUCCESS`. Proto nemusí být jako poslední výraz uveden příkaz `quit`. Na jedné řádce v souboru může být uveden nejvýše jeden výraz. Pokud zadaný soubor neexistuje, pak program vypíše chybovou hlášku „Invalid input file!\n“ a skončí s návratovou hodnotou `EXIT_FAILURE`.

Ukázku spuštění Vašeho programu naleznete na ukázce konzolového rozhraní 1 na straně 3.

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechtě obsahuje všechny zdrojové soubory potřebné k přeložení programu, **Makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `Makefile` a pro Windows `Makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému  $\text{T}_{\text{E}}\text{X}$  ( $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ ). Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

#### Popis činnosti programu

Program zpracuje postupně zadané příkazy a aritmetické výrazy obsahující celočíselné operandy s neomezenou velikostí. Po zadání (ať už z konzole nebo načtením ze souboru) výraz vyhodnotí a jeho výsledek okamžitě vypíše na obrazovku. Chování interpretu (číselnou soustavu výsledků vyhodnocení aritmetických výrazů) bude možné změnit pomocí příkazů uvedených v tabulce 2 na straně 5. Minimální množinu unárních či binárních operátorů, které tento interpret musí zvládnout zpracovat, aby úspěšně prošel validátorem, je uveden v tabulce 3 na straně 6 (z vlastní iniciativy můžete naprogramovat i další operátory či funkce).

<sup>1</sup>Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10/11) a s běžnými distribucemi Linuxu (např. Ubuntu, Debian, Red Hat, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 11 (bullseye) s jádrem verze 5.10.0-32-amd64 a s překladačem gcc 10.2.1-6.

## Specifikace vyhodnocovaných výrazů

Výrazem může být pouze příkaz interpretu, nebo aritmetický výraz v infixové formě. Interpret je **case-insensitive** (tj. nerozlišuje velká a malá písmena). Minimální množinu podporovaných příkazů, resp. operátorů ukazuje tabulka 2, resp. 3. Celočíselné operandy mohou být ve zpracovávaných aritmetických výrazech zadány pomocí

- **dekadické** (bez prefixu),
- **binární** (prefix „0b“),
- nebo **hexadecimální** (prefix „0x“)

číselné soustavy. Pro kódování operandů bude použit výhradně **dvojkový doplňkový kód**. Je tedy nutné dbát na to, že obecně platí nerovnosti:

$$\begin{aligned} 3 &= 0b000\dots00011 = \textcolor{red}{0b011} \neq \textcolor{red}{0b11} = 0b111\dots11111 = -1, \\ 11 &= 0b0001011 = 0x0000b = \textcolor{red}{0x0b} \neq \textcolor{red}{0xb} = 0xffffb = 0b1011 = -5 \text{ a jiné.} \end{aligned} \tag{1}$$

## Specifikace výstupu programu

Program bude vždy vypisovat výsledek právě zadaného aritmetického výrazu či příkazu interpretu. Při načtení seznamu příkazů ze souboru musí výstup odpovídat přesně výstupu spuštění programu bez parametru (interaktivní režim).

Při výpisu výsledků pomocí jejich binární či hexadecimální reprezentace je stejně jako u vstupu použit dvojkový doplňkový kód. Vypsán bude vždy minimální potřebný počet binárních či hexadecimálních cifer – levá zvýrazněná strana nerovnosti (1).

V případě chybného zadání příkazu či aritmetického výrazu vypíše interpret chybu dle tabulky 1 a pokračuje ve zpracování dalšího výrazu. Neuvedené chybové stavy ošetřete dle svého uvážení.

## Užitečné techniky a odkazy

Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy standardní, lze k nim nalézt velké množství dokumentace:

1. **Reprezentace znaménkových čísel**  
[https://en.wikipedia.org/wiki/Signed\\_number\\_representations](https://en.wikipedia.org/wiki/Signed_number_representations)
2. **Bitové pole**  
[https://en.wikipedia.org/wiki/Bit\\_array](https://en.wikipedia.org/wiki/Bit_array)
3. **Algoritmus Shunting yard**  
[https://en.wikipedia.org/wiki/Shunting-yard\\_algorithm](https://en.wikipedia.org/wiki/Shunting-yard_algorithm)
4. **Modul *bitstring* jazyka Python**  
<https://pypi.org/project/bitstring/>
5. **Knihovna *GMP* ← obdobný projekt jen pro ilustraci!**  
<https://gmplib.org/>

**Řešení úlohy je zcela ve Vaší kompetenci** – zvolte takové algoritmy a techniky, které podle Vás nejlépe povedou k cíli. Pokud bude něco nejasného, neváhejte mě kontaktovat.

Některé techniky, jako třeba reprezentace binárního čísla 0b011 polem tří jednobajtových celočíselných hodnot (datový typ char), jsou zcela nevhodné a autor takové implementace musí počítat s penalizací ve formě chyby P205, která je uvedena v chybovníku na webových stránkách předmětu.

# Přílohy

Konzolové rozhraní 1: Ukazka interpretace výrazů programem `calc.exe`. Obsah textového souboru použitého na řádku 18 je ukázán v konzolovém rozhraní 2 na straně 4.

```

1 user@machine:~/pc$ cd semestralka && ls
2 doc src CMakeLists.txt dokumentace.pdf Makefile Makefile.win
3 user@machine:~/pc/semestralka$ make &>/dev/null && ls
4 build doc src calc.exe CMakeLists.txt dokumentace.pdf Makefile Makefile.win
5 user@machine:~/pc/semestralka$ ./calc.exe
6 > 1+0x000001 + 0b01
7 3
8 >
9 > 156
10 156
11 > quit
12 user@machine:~/pc/semestralka$ echo $?
13 0
14 user@machine:~/pc/semestralka$ ./calc.exe /some/non/existing/input/file.txt
15 Invalid input file!
16 user@machine:~/pc/semestralka$ echo $?
17 1
18 user@machine:~/pc/semestralka$ ./calc.exe /some/existing/input/file.txt
19 > 0b101011011101110100101011100011* -157384039439298988989898343/ 0x0fee
20 13295577532879777792226471797318
21 > 0xafadf7868373875afedbcddcbad ^ (-0b101)
22 -1055545516206345285156888947049657551048710594099743746713736133776043059270691
23 813009459988706699
24 > 0xfadf7868373875afedbcddcbad ^ (-0b0101)
25 0
26 > bin
27 bin
28 > (-98587561524232154855 % 0x829874ab3ff7398374fedbacdbacc -0b01011) ^ (0b011 %
29 0x37468237fcdcaabbcc)
30 0b10110011101011000101010100001000011111101000011101110001101011100101010111001
31 0010101110101001110111100111100111101000000110001001110001100001000000101011000
32 1101000011110010000110001110001011010111000
33 > hex
34 hex
35 > -(0xf1!^0b0100101)
36 Input of factorial must not be negative!
37 > -((-0xf1)!^0b0111)
38 0xca255c1f6ecaad7f1c7a66019f01695f9796ce9cc7a3572f7a3a000000000000000000000
39 > -49!^9
40 0xde35af7f5acc4c23f295d9d166bf49d25066f992282373231867c7595bd8d2944cea09acce2abf
41 ea948244a3aefe7434637cd3308e1b0bafd7d248e02dc02b27db03e79c47b733b853fe9be9cca5bd
42 69d6adbb47468f15ee99345dfef44b140b1a222d9ad85b25fb0324307a8d687ce1fd227cc322b4a0
43 70c995446e505f0258781e571206de49d32ebc600e21bfa5880ba49b0cf977e46024945342e4c9b2
44 ce97fd83c07ba2835163ef4a67c54cf3cbec3b0de2a47076400000000000000000000000000000
45 0000000000000000000000000000000000000000000000000000000000000000000000000000
46 user@machine:~/pc/semestralka$ echo $?
47 0
48 user@machine:~/pc/semestralka$

```

Tabulka 1: Minimální množina ošetřených chybových stavů a odpovídajících hlášení.

Popis chyby	Ukázka chybového hlášení
Na vstup interpretu byl zadán neznámý příkaz, tj. takový, který není uveden v tabulce 2.	<pre>&gt; out dec &gt; some_missing_command Invalid command "some_missing_command"! &gt;</pre>
Ve vstupním aritmetickém výrazu je syntaktická chyba.	<pre>&gt; 1+ 1 2 &gt; 1*+ (549818351954^546812!) Syntax error! &gt;</pre>
Při výpočtu došlo k dělení nulou.	<pre>&gt; 5939875389* 0b10010101/ (0x1 -0b01) Division by zero! &gt;</pre>
Výpočet faktoriálu záporného čísla.	<pre>&gt; 0b11100101010011! Input of factorial must not be negative! &gt;</pre>

Konzolové rozhraní 2: Obsah vstupního souboru z ukázky konzolového rozhraní 1 (řádek 18).

```

1 user@machine:~/pc/semestralka$ cat /some/existing/input/file.txt
2 0b101011011101110100101011100011* -157384039439298988989898343/ 0x0fee
3 0xafadf7868373875afedbcddcbad ^ (-0b101)
4 0x0fadf7868373875afedbcddcbad ^ (-0b0101)
5 bin
6 (-98587561524232154855 % 0x829874ab3ff7398374fedbacdbacc -0b01011) ^ (0b011 % 0x
7 37468237fcdaabbcc)
8 hex
9 -(0xf1!^0b0100101)
10 -((-0xf1)!^0b0111)
11 -49!^9
user@machine:~/pc/semestralka$

```

Tabulka 2: Minimální množina podporovaných řídicích příkazů interpretu.

Příkaz	Význam	Příklad
dec	Výsledky vyhodnocení dále zadaných aritmetických výrazů budou vypisovány v desítkové soustavě ( <b>výchozí nastavení</b> ).	<pre>&gt; dec dec &gt; 1+1 2 &gt;</pre>
bin	Výsledky vyhodnocení dále zadaných aritmetických výrazů budou vypisovány ve dvojkové soustavě (standardně s prefixem 0b).	<pre>&gt; bin bin &gt; 1+1 0b010 &gt;</pre>
hex	Výsledky vyhodnocení dále zadaných aritmetických výrazů budou vypisovány v šestnáctkové soustavě (standardně s prefixem 0x).	<pre>&gt; hex hex &gt; 1+1 0x2 &gt;</pre>
out	Vypíše aktuálně používanou číselnou reprezentaci výsledků interpretu (dec, bin nebo hex).	<pre>&gt; out hex &gt; dec dec &gt; out dec &gt; 1+ 1 2 &gt;</pre>
quit	Ukončí interpret s návratovou hodnotou EXIT_SUCCESS.	<pre>&gt; quit user@machine:~/pc/semestralka\$ echo \$? 0 user@machine:~/pc/semestralka\$</pre>

Tabulka 3: Minimální množina podporovaných operátorů a funkcí.

Operátor	Význam	Precedence	Arita	Asociativita	Příklad	Výsledek
!	Vrátí hodnotu faktoriálu zadaného operandu.	4	unární	levá	-0x0c!	-479001600
^	Vrátí výsledek umocnění základu exponentem.	4	binární	pravá	-0b010^3!	-64
-	Vrátí číslo opačné k zadanému operandu.	3	unární	pravá	-0b10001000	120
*	Vynásobí operandy a vrátí výsledek.	3	binární		0b100111*4	-100
/	Vydělí od sebe operandy a vrátí výsledek. Prováděno bude vždy jen celočíselné dělení.	3	binární	levá	0x19/0b0101	5
%	Vrátí zbytek po celočíselném dělení operandů, pro který platí výraz $a \% m = a - m \lfloor \frac{a}{m} \rfloor$ , kde $\lfloor \frac{a}{m} \rfloor$ označuje směrem k nule zaokrouhlený podíl $\frac{a}{m}$ . Tuto variantu implementují i překladače jazyka ANSI C.	2	binární	levá	-10%0b011	-1
+	Sečte operandy a vrátí výsledek.	1	binární		315+0b10011100	215
-	Odečte operandy a vrátí výsledek.	1	binární	levá	315-0b10011100	415
(		0				
)		0				