ZALG 13. cvičení

Celočíselné datové typy v c++

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-9223372036854775808 to 9223372036854775807
signed long int	8bytes	same as long int
unsigned long int	8bytes	0 to 18446744073709551615
long long int	8bytes	-(2^63) to (2^63)-1
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615

Počítání s velkými čísly

- Chceme počítat s číslem, na který nám nestačí rozsah (číslo je větší jak 64 bitů)
- Potřebujeme vlastní datovou strukturu

```
const size_t N = 200;

class BigInt {
private:
    enum SIGN : bool {
        POSITIVE, NEGATIVE
    };
    std::vector<int> digits;
    bool sign;
    unsigned int size;
```

Konstruktory

```
BigInt() {
    digits = std::vector<int>(N,  value: 0);
    sign = POSITIVE;
    size = 0;
}

explicit BigInt(const std::string & number) : BigInt() {
    size_t length = number.length();
    int sub = 0;
    if(number[0] == '-') {
        length--;
        sign = NEGATIVE;
        sub = 1;
    }

    size = length;

    unsigned int index = N - length;
    for(char digit : number.substr( pos: sub)) {
        digits[index] = digit - '0';
        index++;
    }
}
```

Výpis

```
void print() {
    if (sign == NEGATIVE) {
        std::cout << "-";
    }

    for(unsigned int i = N - size; i < N; i++) {
        std::cout << digits[i];
    }

    std::cout << std::endl;
}
```

Sčítání velkých čísel

$$725 + 456 = (700 + 400) + (20 + 50) + (5 + 6) = (7 + 4) \cdot 100 + (2 + 5) \cdot 10 + (5 + 6)$$

	11	**7**	**11**
	11	**7 + 1 = 8**	**1**
$725 + 456 =$	**1**	**1**	**8**	**1**

Odčítání velkých čísel

$$725 - 456 = (700 - 400) + (20 - 50) + (5 - 6) = (7 - 4) \cdot 100 + (2 - 5) \cdot 10 + (5 - 6)$$

	3	**-3**	**-1**
	3	**-3 - 1**	**-1 + 10**
	3 - 1	**-4 + 10**	**9**
$725 - 456 =$	**2**	**6**	**9**

Násobení velkých čísel

$$725 \times 456 =$$
$$= (700 \times 456) + (20 \times 456) + (5 \times 456) =$$
$$= (700 \times 400) + (700 \times 50) + (700 \times 6) + (20 \times 400) + (20 \times 50) + (20 \times 6) + (5 \times 400) + (5 \times 50) + (5 \times 6) =$$
$$= [(700 \times 400) + (20 \times 400) + (5 \times 400)] + [(700 \times 50) + (20 \times 50) + (5 \times 50)] + [(700 \times 6) + (20 \times 6) + (5 \times 6)] =$$
$$= [(700 \times 6) + (20 \times 6) + (5 \times 6)] + [(700 \times 50) + (20 \times 50) + (5 \times 50)] + [(700 \times 400) + (20 \times 400) + (5 \times 400)] =$$

1) $[(7 \times 6) \times 100 + (2 \times 6) \times 10 + (5 \times 6)]$

			42	**12**	**30**
			42	**12 + 3 = 15**	**0**
			42 + 1 = 43	**5**	**0**
		4	**3**	**5**	**0**

Násobení velkých čísel

$[(700 \times 6) + (20 \times 6) + (5 \times 6)] + [(700 \times 50) + (20 \times 50) + (5 \times 50)] + [(700 \times 400) + (20 \times 400) + (5 \times 400)]$

2) $[(7 \times 6) \times 100 + (2 \times 6) \times 10 + (5 \times 6)] + [(7 \times 5) \times 100 + (2 \times 5) \times 10 + (5 \times 5)] \times 10$

		4 + 35	**3 + 10**	**5 + 25**	**0**
		39	**13**	**30**	**0**
		39	**13 + 3 = 16**	**0**	**0**
		39 + 1 = 40	**6**	**0**	**0**
	4	**0**	**6**	**0**	**0**

Násobení velkých čísel

3) $[(7 \times 6) \times 100 + (2 \times 6) \times 10 + (5 \times 6)] + [(7 \times 5) \times 100 + (2 \times 5) \times 10 + (5 \times 5)] \times 10$
$+ \mathbf{[(7 \times 4) \times 100 + (2 \times 4) \times 10 + (5 \times 4)] \times 100}$

	4 + 28	**0 + 8**	**6 + 20**	**0**	**0**
	32	**8**	**26**	**0**	**0**
	32	**8 + 2 = 10**	**6**	**0**	**0**
	32 + 1 = 33	**0**	**6**	**0**	**0**
$725 \times 456 =$	**3**	**3**	**0**	**6**	**0**	**0**

7.2.5 Násobení celých čísel

Zde nám postačí umět vynásobit nezáporná celá čísla. V soustavě se základem z hledáme součin dvou čísel $u = u_1 u_2 \ldots u_n$ a $v = v_1 v_2 \ldots v_m$. Výsledek bude

$$u \times v = w = w_1 w_2 \ldots w_{m+n}.$$

Všimněte si, že nyní nepředpokládáme stejný počet cifer v obou činitelích.

Vyjdeme opět od tradičního násobení na papíře. Při tomto všeobecně známém postupu jsme napočítávali parciální součiny a celkový výsledek jsme získali jako jejich součet. Při strojovém výpočtu bude výhodnější ihned přičítat jednotlivé číslice parciálních součinů k celkovému výsledku.

Algoritmus bude opět využívat pomocnou proměnnou p pro přenos; i a j slouží jako parametry cyklů.

1. *Inicializace*: Položíme $w_{m+1} := 0, \ldots, \; w_{m+n} := 0, j := m$.
2. *Test nuly*: Je-li $v_j = 0$, nastavíme $w_j := 0$ a jdeme na bod 6. (Tento krok lze vynechat.)
3. *Inicializace i*: $i := n$, $p := 0$.
4. *Násobení a sčítání*: Položíme $t := u_i \times v_i + w_{i+j} + p$. Dále položíme $w_{i+j} := t \bmod z$, $p := [t/z]$. Symbol $[x]$ zde znamená celou část čísla x. (Přenos bude vždy v rozmezí $0 \leq p < z$, tedy jednociferný.)
5. *Konec cyklu podle i*: Zmenšíme i o jedničku; je-li $i > 0$, vrátíme se na bod 4, jinak položíme $w_j := p$.
6. *Konec cyklu podle j*: Zmenšíme j o jedničku; je-li nyní $j > 0$, vrátíme se na bod 2, jinak konec.

Násobení velkých čísel – Karatsubův algoritmus

- Mějme čísla *x*, *y*, kde
  - $x = a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + a_1 \cdot 10 + a_0$
  - $y = b_n \cdot 10^n + b_{n-1} \cdot 10^{n-1} + b_1 \cdot 10 + b_0$
- Chceme $z = x \cdot y$
- Čísla *x* a *y* můžeme zapsat jako:
  - $x = a \cdot 10^{\frac{n}{2}} + b$
  - $y = c \cdot 10^{\frac{n}{2}} + d$

Př. : $123\,456 = 123 \cdot 10^3 + 456$

Karatsubův algoritmus

- $z = x \cdot y = \left(a \cdot 10^{\frac{n}{2}} + b\right) \times \left(c \cdot 10^{\frac{n}{2}} + d\right) =$
- $= a \cdot 10^{\frac{n}{2}} \cdot c \cdot 10^{\frac{n}{2}} + a \cdot 10^{\frac{n}{2}} \cdot d + b \cdot c \cdot 10^{\frac{n}{2}} + b \cdot d =$
- $= ac \cdot 10^{n} + (ad + bc) \cdot 10^{\frac{n}{2}} + bd$

- Dvojí násobení $ad + bc$ se vypočíst pomocí pouze jednoho násobku a násobků ac a bd (které musíme spočítat tak či tak):
  - $ad + bc = (a + b)(c + d) - ac - bd$

Karatsubův algoritmus

- $x \cdot y = ac \cdot 10^n + \big((a+b)(c+d) - ac - bd\big) \cdot 10^{\frac{n}{2}} + bd$
- Součin 2 čísel o *n* číslicích dokážeme spočítat přes 3 součiny čísel o $\frac{n}{2}$ číslicích (+ 2 součty a 2 rozdíly čísel o $\frac{n}{2}$ číslicích a 3 součty čísel o „n“ číslicích)
- Z Master Theorem dostaneme pro rekurentní vztah složitost:
  - $T(n) = 3 \cdot T\left(\frac{n}{2}\right) + O(n)$ (součet čísel je $O(n)$ operace)
  - $a = 3, b = 2, c = 1 \rightarrow a > b^c \rightarrow typ\ C \rightarrow T(n) = O\big(n^{\log_b a}\big) = O(n^{\log_2 3})$

Dělení čísel

- Knuth‘s algorithm D
- https://ridiculousfish.com/blog/posts/labor-of-division-episode-iv.html

- Jiný způsob: Školácké dělení velkých čísel

Reprezentace desetinného čísla celým číslem

7	2	5	6	4	3	2

Desetinné číslo 725,6432 můžeme reprezentovat pomocí celého čísla, pokud si budeme pamatovat počet desetinných číslic

```
const size_t K = 200;
const size_t M = 100;

class BigDec {
    BigInt number;
    int dec_digits;
```

Reprezentace desetinného čísla celým číslem

Číslo 725,6432 uložené v poli délky 10, kde prvních 5 indexů reprezentuje celou část, a zbylých 5 indexů desetinnou část

Použití Taylorovy věty

- Pro naši datovou strukturu chceme umět i funkce *log, exp, sin, cos, …*
- Pokud máme implementovány operace +, -, x a / můžeme k tomu použít Taylorovy (Maclaurinovy) řady

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \quad \& \quad O = (-\infty, +\infty)$$

$$\ln(1+x) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n} \quad \& \quad O = (-1, 1\rangle$$

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \quad \& \quad O = (-\infty, +\infty)$$

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} \quad \& \quad O = (-\infty, +\infty)$$

Problém: Pomalá konvergence

Věta 1.2.1. (Taylorova) *Nechť existuje okolí H_a bodu a takové, že funkce f v něm má konečnou $(n+1)$-ní derivaci a nechť $x \in H_a$. Pak zbytek v Taylorově vzorci $f(x) = T_n(x) + R_n(x)$ má tvar*

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,(x-a)^{n+1}\,.$$

Číslo ξ závisí na x a n a leží uvnitř intervalu s krajními body x,a.

- Počítám-li Taylorem $\sin(1000)$ může chyba R_n být po 100 krocích až:
  - $R_n \geq \left|\frac{\sin(\xi)}{101!}(100-0)^{101}\right| \rightarrow R_n \geq \frac{1000^{101}}{101!} \approx$ 1.07151e+142
  - Současně už například 20. člen řady je: $\frac{1000^{61}}{61!} \approx$ 3e+73 (takové čísla například ani nemusíme být schopní spočítat naší datovou strukturou, pokud budou mimo náš rozsah)

Konvergence v okolí středu (a = 0)

- Taylorovy řady konvergují velmi rychle v okolí jejich středu
- Příklad:

$$e^{0,5} = \sum_{n=0}^{\infty} \frac{0,5^n}{n!}$$

- Chyba po k krocích je:

$$R_n \geq \left| \frac{e^{\xi}}{n+1!} (0,5-0)^{n+1} \right| \geq 2 \frac{1}{2^{n+1}(n+1)!}$$

- Tedy například chyba po 50 krocích musí být menší než 5.7260504e-82

exp

- e je všeobecně známá konstanta její hodnotu si můžeme například uložit do globální proměnné
- Máme-li číslo $e^{4,75} = e^{0,75} \cdot e \cdot e \cdot e \cdot e$
- Tedy přes Taylorovu řadu spočteme pouze číslo $e^{0,75}$ a pak ho 4x vynásobíme konstantou e

sin a cos

- Sinus a cosinus jsou funkce s periodou 2π. Od čísla *x* tedy můžeme jednoduše odčítat resp. přičítat hodnotu 2π, dokud nedostaneme číslo *y* z intervalu (-π, π)
- Dále víme, že sinus je posunutý cosinus, platí vztah: $\sin(x) = \cos(x - \frac{\pi}{2})$
- Číslo *y* z intervalu (- π, π) dokážeme ještě převézt na číslo *z z intervalu* $(-\frac{\pi}{2}, \frac{\pi}{2})$ opačné funkce
- Hodnotu π se vyplatí mít předpočítanou jako konstantu
- Př. :
$\sin(102) = \sin(102 - 16 \cdot 2\pi) \approx \sin(1{,}469) = \cos\left(1{,}469 - \frac{\pi}{2}\right) = \cos(-0.1017)$

Logaritmus

- Taylorova řada logaritmu konverguje obecně velmi pomalu, neobsahuje totiž ve jmenovateli faktoriál.
- Dá se ovšem využít vhodně faktu $\ln(xy) = \ln(x) + \ln(y)$
- Př: $\ln(100) = \ln\left(\frac{100}{2} \cdot 2\right) =$

$$= \ln\left(\frac{100}{4} \cdot 4\right) = \ln\left(\frac{100}{2^7} \cdot 2^7\right) = \ln(0.78125 \cdot 2^7) = \ln(0.78125) + \ln(2^7) = \ln(0.78125) + 7\ln(2)$$

- Tedy číslo *x* dělíme resp. násobíme 2 do té doby, dokud nedostaneme číslo *y* z intervalu $(0, \overline{6}; 1, \overline{3})$, hodnotu ln(2) se opět vyplatí mít předpočítanou jako konstantu
- Protože $R_n \geq \left|\frac{n! \cdot x^{n+1}}{(n+1)!(1+x)^{n+1}}\right| = \left|\frac{x^{n+1}}{(n+1)(1+x)^{n+1}}\right|$

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n} \ln x = \frac{(n-1)!(-1)^{n-1}}{x^n}$$

Substituce y = 1 + x

Další funkce

- Spoustu dalších funkcí dokážeme odvodit z těchto 4
- $\tan x = \frac{sinx}{cosx}$
- $\operatorname{cotan} x = \frac{cosx}{sinx}$
- $\log_x y = \frac{\ln y}{\ln x}$
- $x^y = e^{y \cdot \ln x}$, pokud $x > 0$