DÚ 3 – Skupiny 06 a 23

Termín odevzdání DÚ (prvního pokusu) je **PÁTEK 5. 12. 23:59**.

Třetí domácí úkol se bude věnovat výpočtu odmocniny. Přečtěte si zadání až do konce, i na poslední stránce se mohou skrývat užitečné informace (třeba ukázky běhu programu). Abych vás donutil procvičit si funkce a rekurzi, vaše řešení úlohy **NESMÍ OBSAHOVAT JAKÝKOLIV CYKLUS (while, for, do while) ANI PŘÍKAZ GOTO A NESMÍ VYUŽÍVAT VESTAVĚNÉ MATEMATICKÉ FUNKCE JAZYKA C ANI KNIHOVNU MATH.H**. Protože je to opravdu důležité, radši to ještě jednou zopakuji:

vaše řešení úlohy **NESMÍ OBSAHOVAT JAKÝKOLIV CYKLUS (while, for, do while) ANI PŘÍKAZ GOTO A NESMÍ VYUŽÍVAT VESTAVĚNÉ MATEMATICKÉ FUNKCE JAZYKA C ANI KNIHOVNU MATH.H**

Nyní již k samotnému problému. Program přečte od uživatele dvě čísla: jaké číslo chceme odmocňovat (**odmocněnec**) a kolikátou odmocninu chceme vypočítat (pro jednoduchost ho nazveme **odmocnitel**). Pro hodnoty platí tyto podmínky:

* Mocněnec musí být kladné reálné číslo (pro záporná čísla není odmocnina definována),
* odmocnitel kladné celé číslo (záporná odmocnina není definována a desetinné odmocniny ignorujeme),
* výsledkem je opět kladné reálné číslo.

Pro nalezení odmocniny použijeme **algoritmus binárního půlení**. Bližší informace naleznete zde:

<https://is.muni.cz/auth/el/1433/podzim2012/IB111/um/cisla.pdf>

Podívejte se na **slide 33** a dále, je tam i algoritmus, kterým se můžete inspirovat (nerekurzivní a pouze pro druhou odmocninu). Jeho funkci zde shrneme znova (podívejte se na obrázek ve výše uvedených materiálech):

Algoritmus pracuje tak, že „hádá“ čísla a ověřuje, jestli se jedná o hledanou odmocninu. Nehádá je ale náhodně. V každém kroku pracuje s nějakým číselným intervalem (tzn. čísly mezi určitou horní a dolní hranicí). V prvním kroku nastaví spodní hranici na **0** a horní na **odmocněnec**. Dále:

1. Určí **kandidáta** na výsledek jako průměr těchto dvou čísel – nalezne tedy střed intervalu;
2. Provede **umocnění** **kandidáta** na **odmocnitele** - Hledáme např. třetí odmocninu ze dvou, takže umocníme kandidáta na třetí, abychom zjistili, jestli je kandidát vyhovující a jeho třetí mocnina se rovná dvěma – viz dále;
3. Porovná ho s původním **odmocněncem** -Vzhledem k tomu, že výpočty s desetinnými čísly v počítači nejsou zcela přesné, není možné kontrolovat přesnou shodu umocněného kandidáta a mocněnce, musíme proto postupovat následovně:
	1. Odečte od sebe **umocněného kandidáta** a **odmocněnce**
	2. Zjistí **absolutní hodnotu** tohoto rozdílu
	3. Pokud je **absolutní hodnota** menší, než nějaká programátorem stanovená hodnota (v našem případě ji nastavíme na **0.001**), prohlásí kandidáta za výsledek, jinak pokračuje dále.
4. Dále mohou nastat dvě situace, na základě kterých omezíme interval vyhledávání a spustíme celý proces od bodu 1 znovu (toto je místo pro rekurzi):
	1. **Umocněný kandidát** je větší než **odmocněnec** – V takovém případě spodní hranici necháme stejnou a horní hranici snížíme na hodnotu kandidáta (všechna vyšší čísla také budou nepoužitelná) a znovu spustíme celý vyhledávací proces;
	2. **Umocněný kandidát** je menší než **odmocněnec** – V takovém případě budou naopak všechna čísla menší než kandidát nepoužitelná, nastavíme proto spodní hranici hledání na hodnotu kandidáta a horní necháme stejnou a znova spustíme celý vyhledávací proces.

V každém kroku algoritmu se bude interval vždy o nějaký kus zmenšovat, až bude tak malý, že se „trefí“ do hledaného čísla. Po určitém počtu kroků (zhruba 10 – 20) se tedy dobere kýženého výsledku.

Pro vás to v domácím úkolu znamená zejména:

* Načíst od uživatele **odmocněnce** (kladný reálný) a **odmocnitele** (kladný celý);
* Ověřit platnost vstupu;
* Vytvořit **rekurzivní** funkci pro binární půlení podle výše popsaného postupu a zavolat ji s výchozím rozsahem intervalu (0, **odmocněnec**);
* Ošetřit správně porovnávání reálných čísel pomocí porovnání **absolutní hodnoty jejich rozdílu** a prahové hodnoty (**0.001**) – stačí nám, že si čísla jsou „dostatečně blízko“
* Vytvořit jednoduchou **rekurzivní** funkci pro umocnění čísla (Nápověda: xn = x \* xn-1 a x0 = 1) – nezapomeňte, že nesmíte používat cykly ani vestavěné funkce;
* Možná si pro přehlednost budete chtít vytvořit i funkci pro výpočet absolutní hodnoty (nezapomínejte, že existující funkce nesmíte použít). V tom případě ji ale nepojmenovávejte **abs**, protože vám název může kolidovat s tou, která je součástí jazyka C.

Program se dá pohodlně napsat na zhruba 60 řádků. Na další stránce vidíte ukázkové běhy programu včetně „ladících“ informací pro každý krok výpočtu – aktuální krok, spodní a horní hranici intervalu a aktuálního kandidáta na výsledek. Tyto výpisy váš program nemusí obsahovat. Ověřte si však, že váš program pro všechny vzorové hodnoty funguje správně. Ve **studijních materiálech našich cvičení** se podívejte na příklady s funkcemi a rekurzí, kde se počítá faktoriál a n-tý prvek Fibbonaciho posloupnosti.







