Projekt 1: Matematické funkce

Program očekává povinně jeden parametry příkazové řádky – název funkce. Povinnost a význam dalších parametrů závisí na zvolené funkci.

Základní vlastnosti programu:

- dodržení normy ISO/IEC 9899:2011
- kontrola vstupů (parametry příkazové řádky, soubory; ***nedodržení –10%***)
- hlášení chyb (výpis hlášky na stderr, návratová hodnota funkce main; ***nedodržení –10%***)
- jakékoliv dodatečné a či ladící informace vypisujte na *stderr* (standardní chybový výstup)
- na *stdout* (standardní výstup) vypisujte pouze v souladu se zadáním (***nedodržení –25%***)
- pro datový typ vstupních hodnot použijte double (64-bit IEEE-754 floating point).
- zdrojové soubory budou odevzdány v archivu formátu ZIP: (***nedodržení –20%***)
  - název souboru `vcp_prj1_123456.zip` , kde 123456 je Vaše osobní VUT číslo
  - všechny zdrojové .cpp soubory musí být všechny v kořenu archivu (ne v podsložce)
  - NEodevzdávejte konfiguraci projektu, binární soubory; POUZE .c a .h
  - **pokud** projekt rozdělíte do více .c souborů, přiložte `Makefile`, návod na konci
  - **pokud** nepřiložíte `Makefile`, musí být program v souboru main.c (stále může používat hlavičkové .h soubory) a pak musí být možné program přeložit příkazem: `gcc main.c -o vcp_prj1 -std=c11`
- v případě pádu programu pro validní vstupy, které nemají vést na chybu, **–20%** .

Funkce druhá odmocnina, název prvního parametru „sqrt“

- hodnota druhého parametru je komplexní číslo *z*, které může být zadáno ve tvarech:
  - *a+bi, a-bi, a, bi, -bi, i, -i*
  - *a, b* jsou reálná čísla ve formátech `%f` nebo `%e`
  - *i* je imaginární jednotka, reprezentovaná přímo znakem `i`
  - příklady: 2.6 -3.47 i -2.679e-6i +44 -19+1.127e2i
- hodnota třetího (nepovinného) parametru *epsilon* určuje maximální odchylku vypočtené hodnoty od skutečné hodnoty
- výchozí hodnota *epsilon* je 1e-5
- zabraňte nekonečnému cyklu
- výstupem je funkční hodnota ve formátu `%.17e%+.17ei`
- OMEZENÍ: je zakázáno použít příslušné matematické funkce ze standardní knihovny, instrict funkce, vložený assembler anebo jiné knihovny

Funkce Mandelbrotova množina, název prvního parametru „mandelbrot“

- hodnota druhého parametru je komplexní číslo *z*, které může být zadáno ve tvarech:
  - *a+bi, a-bi, a, bi, -bi, i, -i*
  - *a, b* jsou reálná čísla ve formátech `%f` nebo `%e`
  - *i* je imaginární jednotka, reprezentovaná přímo znakem `i`
  - příklady: 2.6 -3.47 i -2.679e-6i +44 -19+1.127e2i
- hodnota třetího parametru *n* určuje počet iterací, výchozí hodnota *n* je 75
- můžete využít datových typů a funkcí ze standardní knihovny (complex.h)
- výstupem jsou dvě hodnoty:
  - 0 nebo 1 podle toho jestli bod do Mandelbrotovy množiny patří
  - hodnota $|\mathbf{z_n}|$ po *n* iteracích, formát si můžete vybrat
  - např.: `0 9.64874542`

Funkce skalární součin vektorů, název prvního parametru „dotprod“

- hodnota druhého parametru určuje název/cestu souboru se vstupními daty
    - pokud není zadán, jako vstup se použije *stdin* (standardní vstup)
    - ze souboru (nebo stdin) načtěte dva vektory, každý na jednom řádku; složky vektoru jsou reálná čísla
    - pokud vektory nemají stejné délky, považujte to za chybu
- výstupem je hodnota skalárního součinu ve formátu `%.17e`
- [*bonus ke zkoušce*] druhá varianta dotprod2, která se snaží použít co nejméně operační paměti, ale neumožňuje pracovat se *stdin*, pouze se souborem

Hlášené chyby při ukončení programu:

- žádná chyba – návratový kód 0
- neznámá funkce – návratový kód 1
- nelze interpretovat/načíst hodnotu parametru – návratový kód 2
- nelze otevřít soubor – návratový kód 3
- jiný důvod – návratový kód 4

Příklad Makefile, pouze pokud použijete více .c souborů. V Makefile je nutné používat znak tabulátor pro odsazení příkazů; zde je označen barevně.
(návod: https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/)
Předpokládáme existenci souborů:

- ladik_p1.c
- memecnina.c
- zbytek.c

Funkce main je v souboru ladik_p1.c

Tento Makefile vůbec neřeší závislosti nutné pro zajištění správné rekompilace. Program se stejně bude kompilovat jenom jednou; také nepoužívám makra, která jsou uvedena v návodu v příkladu 3. Pro ještě větší zájemce existují další systémy pro řešení závislostí a kompilace jako Ninja, CMake, premake, SCons, atd.

```
CC=gcc
CFLAGS=-std=c11

all: vcp_prj1

ladik_p1.o:
	$(CC) -c ladik_p1.c -o ladik_p1.o $(CFLAGS)

memecnina.o:
	$(CC) -c memecnina.c -o memecnina.o $(CFLAGS)

zbytek.o:
	$(CC) -c zbytek.c -o zbytek.o $(CFLAGS)

# zde je nutno resit zavislosti spustitelneho souboru na modulech
vcp_prj1: ladik_p1.o memecnina.o zbytek.o
	$(CC) ladik_p1.o memecnina.o zbytek.o -o vcp_prj1 $(CFLAGS)

clean:
	rm -f *.o vcp_prj1
```