Trojúhelníky #2

Termín odevzdání:	**11.11.2018 23:59:59**
Pozdní odevzdání s penalizací:	**06.01.2019 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**4.7667**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	2 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy:	2 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je vytvořit program, který bude porovnávat dvojice trojúhelníků. Úloha je rozšířením jednodušší varianty, umožňuje zadávat vstupní trojúhelníky buď pomocí souřadnic vrcholů nebo pomocí délek stran. Doporučujeme nejprve řešit úlohu jednodušší.

V rovině jsou zadané 2 trojúhelníky. Každý trojúhelník je zadán buď pomocí trojice svých vrcholů, nebo pomocí délek svých stran:

- trojúhelník určený pomocí svých vrcholů je zadán jako trojice souřadnic oddělená čárkami a uzavřená do složených závorek. Každá souřadnice je tvořena dvojicí desetinných čísel v hranatých závorkách, složky souřadnice jsou oddělené středníkem. Např.:

  ```
  { [ 1.5; 2 ], [3;4.2], [ 0.5 ; 0.6 ] }
  ```

- trojúhelník určený pomocí délky svých stran je zadán jako trojice desetinných čísel, tato čísla jsou oddělená čárkami a celá trojice je umístěna do složených závorek. Např.:

  ```
  {2.1, 3, 3.4}
  ```

Program přečte zadané trojúhelníky ze svého standardního vstupu a rozhodne se pro jednu z variant:

- zda zadaný vstup tvoří trojúhelník (zadaná trojice bodů netvoří trojúhelník, pokud body leží na jedné přímce, strany musí splňovat trojúhelníkovou nerovnost),
- zda jsou zadané trojúhelníky shodné,
- zda jsou zadané trojúhelníky liší, ale mají stejný obvod, nebo
- zda jsou zadané trojúhelníky zcela odlišné.

Pokud je vstup neplatný, program to musí detekovat a zobrazit chybové hlášení. Chybové hlášení zobrazujte na standardní výstup (ne na chybový výstup). Za chybu považujte:

- nečíselné zadání souřadnic nebo délek stran (neplatné desetinné číslo),
- chybějící souřadnice nebo délka strany,
- chybějící nebo přebývající oddělovače (složené závorky, hranaté závorky, čárky, středníky).

Ukázka práce programu:

```
Trojuhelnik #1:
{[0;0],[5;0],[2.5;3]}
Trojuhelnik #2:
 { [ 4 ; -1 ] , [ 7 ; 1.5 ] , [ 4 ; 4 ] }
Trojuhelniky jsou shodne.
```

```
Trojuhelnik #1:
{
[
0
;
15
]
,       [       112     ;       0       ]       ,[112;15]}
Trojuhelnik #2:
{96.0,40.0,104.0}
Trojuhelniky nejsou shodne, ale maji stejny obvod.
```

Trojuhelnik #1:
{ 10 , 15 , 10 }
Trojuhelnik #2:
{12,8,17}
Trojuhelnik #2 ma vetsi obvod.

Trojuhelnik #1:
{[0;14.04],[11.2;0],[0;0]}
Trojuhelnik #2:
{[20.16;0],[0;2.7],[20.16;2.7]}
Trojuhelniky nejsou shodne, ale maji stejny obvod.

Trojuhelnik #1:
{[0.00;0.00],[0.24;0.70],[0.24;0.00]}
Trojuhelnik #2:
{0.65,0.78,0.25}
Trojuhelniky nejsou shodne, ale maji stejny obvod.

Trojuhelnik #1:
{51.09,49.49,4.94}
Trojuhelnik #2:
{37.92,50.11,17.49}
Trojuhelniky nejsou shodne, ale maji stejny obvod.

Trojuhelnik #1:
{[0;0],[10;0],[0;10]}
Trojuhelnik #2:
{[0;0],[10;10],[15;15]}
Neplatny trojuhelnik.

Trojuhelnik #1:
{[10;0],[20;1],[25;1.5]}
Neplatny trojuhelnik.

Trojuhelnik #1:
{1,2,3}
Neplatny trojuhelnik.

Trojuhelnik #1:
{[0;0],[999.990;204.330],[899.991;183.897]}
Neplatny trojuhelnik.

Trojuhelnik #1:
{1.923,59.240,61.163}
Neplatny trojuhelnik.

Trojuhelnik #1:
{[1;2],[3;abcd],[7,9]}
Nespravny vstup.

Poznámky:

- Ukázkové běhy zachycují očekávané výpisy Vašeho programu (tučné písmo) a vstupy zadané uživatelem (základní písmo). Zvýraznění tučným písmem je použité pouze zde na stránce zadání, aby byl výpis lépe čitelný. Váš program má za úkol pouze zobrazit text bez zvýrazňování (bez HTML markupu).
- Znak odřádkování (\n) je i za poslední řádkou výstupu (i za případným chybovým hlášením).
- Pro reprezentaci hodnot použijte desetinná čísla typu `double`. Nepoužívejte typ `float`, jeho přesnost nemusí být dostatečná.
- Úlohu lze vyřešit bez použití funkcí. Pokud ale správně použijete funkce, bude program přehlednější a bude se snáze ladit.
- Číselné vstupní hodnoty jsou zadávané tak, aby se vešly do rozsahu datového typu `double`. Referenční řešení si vystačí s číselnými typy `double` a `int`.

- Pro načítání vstupu se hodí funkce `scanf`.
- Při použití funkce scanf se může hodit konverze `%c` a `%c` (s mezerou před vlastním %c). Přesný rozdíl si najděte v manuálu.
- Při programování si dejte pozor na přesnou podobu výpisů. Výstup Vašeho programu kontroluje stroj, který požaduje přesnou shodu výstupů Vašeho programu s výstupy referenčními. Za chybu je považováno, pokud se výpis liší. I chybějící nebo přebývající mezera/odřádkování je považováno za chybu. Abyste tyto problémy rychle vyloučili, použijte přiložený archiv se sadou vstupních a očekávaných výstupních dat. Podívejte se na videotutoriál (materiály -> cvičebnice -> video tutoriály), jak testovací data použít a jak testování zautomatizovat.

- Váš program bude spouštěn v omezeném testovacím prostředí. Je omezen dobou běhu (limit je vidět v logu referenčního řešení) a dále je omezena i velikost dostupné paměti (ale tato úloha by ani s jedním omezením neměla mít problém). Testovací prostředí dále zakazuje používat některé "nebezpečné funkce" -- funkce pro spouštění programu, pro práci se sítí, ... Pokud jsou tyto funkce použité, program se nespustí. Možná ve svém programu používáte volání:

```
int main ( int argc, char * argv [] )
{

  ...

  system ( "pause" ); /* aby se nezavrelo okno programu */
  return 0;
}
```

  Toto nebude v testovacím prostředí fungovat - je zakázáno spouštění jiného programu. (I pokud by se program spustil, byl by odmítnut. Nebyl by totiž nikdo, kdo by pauzu "odmáčkl", program by čekal věčně a překročil by tak maximální dobu běhu.) Pokud tedy chcete zachovat pauzu pro testování na Vašem počítači a zároveň chcete mít jistotu, že program poběží správně, použijte následující trik:

```
int main ( int argc, char * argv [] )
{

  ...

#ifndef __PROGTEST__
  system ( "pause" ); /* toto progtest "nevidi" */
#endif /* __PROGTEST__ */
  return 0;
}
```

- Slovní popis struktury platných vstupních dat není zcela exaktní. Proto připojujeme i formální popis vstupního jazyka v EBNF:

```
input      ::= { whiteSpace } triang { whiteSpace } triang { whiteSpace }
triang     ::= '{' (byCoord | bySides) '}'
byCoord    ::= {whiteSpace} '[' coord ']' {whiteSpace} ','
               {whiteSpace} '[' coord ']' {whiteSpace} ','
               {whiteSpace} '[' coord ']' {whiteSpace}
bySides    ::= {whiteSpace} decimal {whiteSpace} ','
               {whiteSpace} decimal {whiteSpace} ','
               {whiteSpace} decimal {whiteSpace}
whiteSpace ::= ' ' | '\t' | '\n' | '\r'
coord      ::= { whiteSpace } decimal { whiteSpace } ';' { whiteSpace } decimal { whiteSpace }
decimal    ::= [ '+' | '-' ] integer [ '.' integer [ ( 'e' | 'E' ) [ '+' | '-' ] integer ] ]  |
               [ '+' | '-'  ] '.' integer [ ( 'e' | 'E' ) [ '+' | '-' ] integer ]
integer    ::= digit { digit }
digit      ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

Vzorová data: **Download**

☐ **Referenční řešení**

2	**04.11.2018 21:29:58**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	4.7667	

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Základní test s parametry podle ukázky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Max doba běhu: 0.006 s (limit: 1.000 s)
    - Celková doba běhu: 0.062 s
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test mezních hodnot': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Max doba běhu: 0.005 s (limit: 1.000 s)
    - Celková doba běhu: 0.158 s
    - Úspěch v nepovinném testu, hodnocení: 100.00 %
  - Test 'Test ošetření nesprávných vstupů': Úspěch
    - Dosaženo: 86.67 %, požadováno: 50.00 %
    - Max doba běhu: 0.007 s (limit: 1.000 s)
    - Celková doba běhu: 0.479 s
    - Úspěch v nepovinném testu, hodnocení: 86.67 %
    - ☐ Nesprávný výstup
    - Nesprávný výstup **[Zpřístupnit nápovědu (193 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (196 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (210 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (194 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (224 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (223 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (224 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (224 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (221 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (152 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (154 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (185 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (187 B)]**
  - Test 'Test náhodnými daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Max doba běhu: 0.007 s (limit: 1.000 s)
    - Celková doba běhu: 0.111 s
    - Úspěch v nepovinném testu, hodnocení: 100.00 %
  - Celkové hodnocení: 86.67 % (= 1.00 * 1.00 * 0.87 * 1.00)
- Použité nápovědy: 1
- Penalizace za vyčerpané nápovědy: Není (1 <= 2 limit)
- Celkové procentní hodnocení: 86.67 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 0.87 * (5.00 + 0.50) = 4.77

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**1**	--	--	--
Řádek kódu:	**142**	**142.00 ± 0.00**	**142**	`main`
Cyklomatická složitost:	**22**	**22.00 ± 0.00**	**22**	`main`

1	**04.11.2018 21:19:13**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - ☐ Chyba při základní kompilaci
- Celkové procentní hodnocení: 0.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 0.00 * (5.00 + 0.50) = 0.00