



mała delta

Czy komputer zna się na matematyce?

Myślisz, że komputer potrafi liczyć? – Możesz się przeliczyć!

Wojtek Pytalski jest zachwycony możliwościami komputerów. Nie widzi nic dziwnego w tym, że używa się ich wszędzie, od dziecięcego pokoju, przez poważne instytucje finansowe, aż po mury wyższych uczelni. Jest przekonany, że są niezawodne i znacznie sprawniejsze w rachunkach niż człowiek i z pewnością nie mylą się w obliczeniach tak, jak to niejednokrotnie zdarza się (zawsze nieco roztargnionym) matematykom.

Czy komputer potrafi liczyć?

Starszy brat, który już od dwóch lat studiuje matematykę, z politowaniem popatrzył na Wojtkę. „Nie ma czym się zachwycać” – ziewnął. – „Zrób test Kahana: komputer nawet nie umie dokładnie obliczyć, ile to jest $(4/3 - 1) \cdot 3 \dots$ ”.



Wojtek spojrział z niedowierzaniem – czy czasem brat nie chce z niego znowu zadrwić? – więc niewiele myśląc, szybko zasiadł do klawiatury, aby przekonać się, jak jest naprawdę. W szkole nauczyli się programować w języku C, więc bez trudu uruchomił następujący programik:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     float x;
6
7     x = (4.0/3.0-1.0)*3.0;
8     printf("Wynik: \n%f\n", x);
9
10    return(0);
11 }
```

Wojtek już wiedział, że gdyby napisał po prostu $x = (4/3-1)*3$, dostałby w wyniku 0 (a czy Ty wiesz dlaczego?). Uruchomił szybciotko program i triumfalnie oznajmił starszemu bratu: „Nieprawda, nie dałem się nabrać: co jak co, ale takie proste obliczenia komputer wykonuje bezbłądnie! Zresztą, sam popatrz na monitor:”

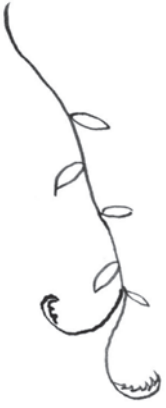
Wynik: 1

„Ech, maluchu, maluchu...” – z politowaniem westchnął brat – „lepiej sprawdź, czy $(4/3 - 1) \cdot 3 - 1$ wyjdzie Ci zero...”. Wojtek zastanowił się – bo niby na czym miałyby polegać różnica? – ale szybko zmienił siódmą linię programu na

```
x = (4.0/3.0-1.0)*3.0-1.0;
```

i uruchomił:

Wynik: -2.22045e-16



Gdy zobaczył ten wynik, zawstydził się, gdyż natychmiast zrozumiał, co przeoczył poprzednio i dlaczego zamiast zera pojawiło się $-2.22045 \cdot 10^{-16}$. No tak, zupełnie zapomniał o tym (a przecież uczyli się tego kiedyś na informatyce), że wyniki obliczeń komputera są zaokrąglane z pewną dokładnością. Na pewno komputer nie jest w stanie obliczyć z absolutną precyzją wyniku dzielenia $4/3 = 1,333333\dots$ (liczba $4/3$ ma nieskończone rozwinięcie, więc trzeba je w pewnym momencie „urwać”). W takim razie i pozostałe obliczenia cierpią z powodu tego zaokrąglenia! Dlatego dla komputera $4/3 - 1$ jest tylko *prawie* równe $1/3$ i w konsekwencji $(4/3 - 1) \cdot 3$ jest tylko *prawie* równe 1; to „prawie”, jak wynika z testu, jest równe mniej więcej $2 \cdot 10^{-16}$ – jest więc bardzo małe, ale nie zerowe.

„Gdyby więc pozbyć się dzielenia i wykonywać obliczenia wyłącznie na zadanych, prostych liczbach, komputer musi dać poprawny wynik!” – wrzasnął uradowany tak głośno, że dotychczas spokojnie śpiący Mruczek zeskoczył ze swojego ulubionego miejsca na regale. „No, staruszk, popatrz!” – odwrócił się do brata – „Wystarczy obliczać

$$(1,1 - 1) \cdot 10 - 1 = 0$$

– w tym wyrażeniu wszystkie liczby są tak proste, że nie ma mowy o błędach...”. W okamgnieniu jeszcze raz zmienił siódmą linijkę swojego kodu na

$$x = (1.1-1)*10-1.0;$$

i ponownie uruchomił program:

Wynik: 8.88178e-16

Bracia popatrzyli po sobie skonsternowani... Co się dzieje?! Dalej niedokładnie?! Starszy szybko wyciągnął z szuflady wysłużony kalkulator (po ostatnim kolokwium z większym krytycyzmem podchodził do swoich możliwości rachunkowych) i wstukał na klawiaturze:



Wyszło równo ZERO.

Więc jak to jest – pomyślał Wojtek – ich wspólny wypasiony komputer z najnowszym procesorem Intela nie umie wykonać bezbłędnie nawet tak banalnego działania, które z drugiej strony ani dla nich samych – ani dla leciwego, dziesięcioletniego kalkulatora – nie jest żadnym problemem? Fakt, na komputerze błąd dalej był malutki – około 0,0000000000000009, ale... dlaczego w ogóle tam był *jakikolwiek* błąd?!

Wszystko wskazywało na to, że – w przeciwieństwie do starego kalkulatora – komputer nie używał dokładnej wartości liczby 1,1, tylko jakiegoś jej przybliżenia... Może więc odkryli poważny defekt w konstrukcji procesora?

Z wypiekami na twarzy zaczęli bardziej systematyczne badania. Wojtek zajął się dalszymi eksperymentami. Starszy brat sięgnął po fachową literaturę i wkrótce natrafił na informację, że liczba 1,1 ma *nieskończone* rozwinięcie dwójkowe. Rzeczywiście, przecież łatwo sprawdzić, sumując szeregi geometryczne, że

$$1,1 = 1 + \sum_{i=1}^{\infty} \left(\frac{1}{2^{4i}} + \frac{1}{2^{4i+1}} \right).$$

To zaś znaczy, że liczba 1,1 zapisana w systemie dwójkowym ma postać:

$$(1,000\ 1100\ 1100\ 1100\ 1100\ \dots)_2.$$





Rozwiązanie zadania F 735.
Zapisując prawo załamania po kolei dla
każdej płytki, otrzymujemy w końcu
kąąt padania odpowiadający kątowii
całkowitego wewnętrznego odbicia.
Wtedy:

$$\sin \alpha_{\min} = n/k^{N-1},$$

zatem $\alpha_{\min} = \arcsin(n/k^{N-1})$.



Komputer przechowuje liczby właśnie w systemie dwójkowym i oczywiście nie jest w stanie zachować w pamięci *nieskończenie wielu* cyfr rozwinięcia! Dokładniej, procesor ich komputera, pracujący pod kontrolą 64-bitowego systemu operacyjnego Linux, może pamiętać jedynie 52 cyfry po przecinku binarnego rozwinięcia liczby 1,1 – a więc wszystkie działania prowadzi nie na 1,1 (jak wcześniej myśleli), ale na liczbie

$$(1,\underbrace{000\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 11010}_{52\ \text{cyfry}})_2$$

– która różni się od wyjściowej wartości 1,1 o około 10^{-16} . Dalszego ciągu łatwo się domyślić.

Tak więc rzeczywiście, nie tylko liczba $4/3$, ale też 1,1 jest reprezentowana w sposób przybliżony w procesorze komputera... Z drugiej strony, kalkulatory zwykle korzystają z reprezentacji liczb w systemie dziesiętnym (a nie dwójkowym) – właśnie dlatego, by nie sprawiać ludziom niespodzianek podobnych do powyższych – i to tłumaczy dokładny wynik obliczenia $(1,1 - 1) \cdot 10 - 1$ na kalkulatorze.

Szukając więcej informacji na ten temat, Wojtek dowiedział się, że nie tylko oni padli ofiarą nadmiernej ufności w precyzję komputerowych obliczeń. To właśnie fałszywe przekonanie wojskowych inżynierów, że komputer (wystarczająco) dokładnie wykonuje obliczenia na całkowitych wielokrotnościach liczby 0,1, było przyczyną zagadkowego zachowania się amerykańskiego systemu antyrakietowego Patriot, który po kilkudziesięciu godzinach bezczynności (tak!) kompletnie tracił celność, co swego czasu umożliwiło skuteczny atak rakietowy wojsk irackich podczas operacji „Pustynna Burza”.

Małą Deltę przygotował Piotr KRZYŻANOWSKI