```
\documentclass[12pt,a4paper]{article}
\title{Raport z Laboratorium 1 - Obliczenia naukowe }
\usepackage{polski}
\usepackage[utf8]{inputenc}
\usepackage[polish]{babel}
\author{Bartłomiej Sadowski}
\begin{document}
\maketitle
\section{zad. 1}
\subsection{maheps}

Zadanie polega na iteracyjnym wyznaczeniu epsilona maszynowego(maheps) dla Float16,
Float32,
Float64 i porównaniu wyników z funkcjami eps(Float*) oraz z danymi z pliku
nagłówkowego języka C.

\begin{tabular}{cccc}
\
  &Float16 & Float32 & Float64 \\
\
Juliat(iteracyjnie)
& 0.00097656
& 1.1920929e-7
& 2.220446049250313e-16 \\
\
Juliat(eps())
 & 0.00097656
 & 1.1920929e-7
 & 2.220446049250313e-16 \\
\
C
 & 0.00097656
 & 1.19209E-07
 & 2.220446e-16
 \\
\
\end{tabular}
Liczba $maheps$ (machine epsilon) jest to taka liczba, że $1.0 + maheps > 1.0$
gdzie $macheps > 0$
i jest najmniejszy z możliwych wartości, większych od 0.

W tabelce powyżej mamy zawarte wartości epsilonu maszynowego dla poszczególnych
prezycji wyznaczonych iteracyjnie. Sama procedura polega na dzieleniu wartości$
x=1.0 przez 2$ dopóki $x+1.0>1.0$ i zwróceniu przedostatniej wartości.

\subsection{eta}

Zadanie polega na iteracyjnym wyznaczeniu liczby $eta$ czyli najmniejszej liczby,
takiej że $eta>0.0$ dla Float16, Float32, Float64 i porównaniu wyników z funkcjami
nextfloat(float*(0,0)) oraz z danymi z pliku nagłówkowego języka C.

\begin{tabular}{cccc}
\
  &Float16 & Float32 & Float64 \\
\
Juliat(iteracyjnie)
& 5.9605e-8
& 1.0e-45
& 5.0e-324 \\
\
Juliat(nextfloat(float*(0,0))
 & 5.9605e-8
```

```latex
 & 1.0e-45
 & 5.0e-324 \\
\
C
 & 5.9605e-8
 & 1.0e-45
 & 5.0e-324
 \\
\
\end{tabular}

Procedura wyznaczania nextfloat analogiczna do sposobu wyznaczania maheps z tą
różnica że zaczynamy od x=1.0 i dzielimy przez 2 do osiągnięcia stanu $x<0.0$

\subsection{MAX}

Zadanie polega na iteracyjnym wyznaczeniu maksymalnej wartości $(MAX)$ dla Float16,
Float32,Float64 i porównaniu wyników z funkcjami realmax(Float*) oraz z danymi z
pliku nagłówkowego języka C.

\begin{tabular}{cccc}
\
  &Float16 & Float32 & Float64 \\
\
Juliat(iteracyjnie)
& 1.7976931348623157e308
& 3.4028235e38
& 65504.0 \\
\
Juliat(realmax(*))
& 1.7976931348623157e308
& 3.4028235e38
& 65504.0 \\
\
C
 & 1.797693e+308
 & 3.4028234664e+38
 & 65504.0
 \\
\
\end{tabular}

Powyższa tabelka zawiera zestawienie maksymalnych wartości typów
zmiennoprzecinkowych.
    Iteracyjny sposób wyznaczania wygląda następująco. Na początku wypełniana jest
mantysa całej zmiennej liczbą $\frac{1}{2^k}$ gdzie $k \in \lbrace 1,2 \ldots 53
\rbrace$. Następnie zmienna mnożona
jest przez  $2$ aż do osiągnięcia stanu $inf$.

\subsection{wnioski}
\subsubsection{Związek maheps z precyzją arytmetyki $\varepsilon$}
$\varepsilon - $ precyzja arytmetyki, to właśnie maheps. Zachodząca nierówność
$\delta \leq
\varepsilon $ oznacza iż błąd względny jednej operacji liczb maszynowych będzie nie
większy niż
maheps. Z powyższego zdania, wynika iż nasze obliczenia zależą w głównej mierze od
ilości bitów dostępnej dla mantysy, gdzie im większa ich liczba tym dokładniejszy
wynik.

\subsubsection{Związek eta z liczbą $MIN_{sub}$}
$MIN_{sub}$ a więc eta, jest to najmniejsza liczba będąca w otoczeniu 0. Reszta
liczb $|x|$ jest reprezentowana jako $0^-$ niedomiar. Mantysa danej liczby jest
```

```
zbyt mała aby można było ją przedstawić w danej arytmetyce.

\subsubsection{$realmax$}
$realmax$ to oczywiście maksymalna wartość jaką może przyjąć liczba w danej
precyzji

\section{zad. 2}
Zadanie polega na eksperymentalnym pokazaniu, iż formuła $3*(4/3-1)-1$ daje w
wyniku epsilon maszynowy.

Wyniki dla równania $3*(4/3-1)-1$\\


\begin{tabular}{ccc}

Float64 & Float32 & Float16 \\

-2.220446049250313e-16 & -2.220446e-16 & -0.0 \\

\end{tabular} \\


Otrzymane rezultaty wynikają z błędów zaokrągleń. Aby zilustrować to dokładniej,
wypiszemy kolejne działania dla Float64\\


$(4/3-1)=0.33333333333333326$ pierwsze działanie to odjęcie od nieskończonego
binarnie ułamka okresowego liczby 1 który daje nam nawarstwiony błąd \\\

$0.33333333333333326*3 = 0.9999999999999998$ kolejny krok to zbliżenie się wynikiem
do jedynki\\

$0.9999999999999998-1 = -2.220446049250313e-16$ a na końcu od wyniku odjęcie
jedynki, co w wyniku powoduje wpadnięcie wyniku w maheps a więc niedomiar\\
\section{zad. 3}
Zadanie polega na pokazaniu, iż liczby w przedziale $[1,2]$ są równomiernie
rozmieszczone z krokiem
$\delta =2^{-52}$ a każdą liczbę $x$ można zaprezentować w postaci $x=1+k*\delta$
gdzie $k=2^{0},2^{1},2^{2}\ldots,2^{52}-1$.
Aby pokazać, iż tak się dzieję użyję $k=4$ gdzie $n$ to następnik a $p$ to
poprzednik danego $k$:\\\
\begin{tabular}{|c|c|c|}
\hline
$k-1$ & 0011111111110000000000000000000000000000000000000000000000001000 \\
\hline
$k$ & 0011111111110000000000000000000000000000000000000000000000010000 \\
\hline
$k+1$ & 0011111111110000000000000000000000000000000000000000000000100000 \\
\hline
\end{tabular} \\\
Z powyższej tabelki wynika, że dla kolejnych wartości k liczby we Float64 różnią
się tylko w jednym miejscu dla przedziału $[1,2]$.

Dla przedziału $[1/2],[1]$ wzór wygląda następująco $x=\frac{1}{2}+k*\delta$ a
wyniki dla przykładowego $k=49$ wyglądają następująco:
\begin{center}

\begin{tabular}{|c|c|c|}
\hline
$k-1$ &0011111111100010000000000000000000000000000000000000000000000000 \\
\hline
$k$   &0011111111100100000000000000000000000000000000000000000000000000 \\
```

```
\hline
$k+1$ &0011111111101000000000000000000000000000000000000000000000000000 \\
\hline
\end{tabular} \\\
\end{center}

Dla przedziału $[2],[4]$ wzoru ogólnego nie da się wyprowadzić gdyż największą
wartością jaką można uzyskać w samej mantysie będzie < od 1
$(2^{52}-1)*(2^{-52})=(1-2^{-52}) < 1 $

\section{zad. 4}

Zadanie polega na iteracyjnym odnalezieniu liczby dla której wyrażenie $ x*\frac{1}
{x} \neq x $

Oczywiście najmniejszą taką liczbą będzie maheps który po podstawieniu za x daje
wynik $1.000000002$
Błąd wynika z błędów zaokrągleń . Gdy mianownik w porównaniu do licznika różni się
znacząco wynik zostaje zaokrąglony na zasadzie $round to even$

\section{zad. 5}
Zadanie polega na obliczeniu iloczynu wektorowego wektorów $x$ i $y$ za pomocą
różnych algorytmów. \\

$x=  [2.718281828,-3.141592654,1.414213562,0.5772156649,0.3010299957]$\\

$y=  [1486.2497,878366.9879,-22.37492,4773714.647,0.000185049]$
\begin{center}
\begin{tabular}{|l|p{3cm}|p{3cm}|p{2cm}|p{2cm}|p{3cm}|}
 \hline
 & a & b & c & d & wynik  \\
 \hline
 F64 &
 1.56433088
 70494366e(-10) &
-1.56433088
70494366e(-10) &
 0.0 &
-2.759502
9196180
594e6 &
-1.00657107
00000010e-11 \\
 \hline
 F32 &
 -0.4999443 &
 -0.4543457 &
 -0.3929138
 18359375 &
 -0.5 &
  x  \\
 \hline
 \end{tabular}
 \end{center}

Na podstawie powyższego przykładu widać jak bardzo w obliczeniach
zmiennoprzecinkowych należy rozsądnie wybierać algorytm gdyż różnice w wynikach
mogą okazać się wręcz kolosalne.
\section{zad. 6}
Zadanie polega na obliczeniu kolejnych wartości funkcji $f(x) = \sqrt{x^{2}+1}-1$ i
$g(x)=\frac{x^{2}}{\sqrt{x^{2}+1}+1}$ dla $x = 8^{-1},8^{-2}... $
```

```latex
Przy $(\frac{1}{x})^{9} $ funkcja $f(x)$ zaokrągla wyniki do 0. \\


\begin{tabular}{|c|c|}
\hline
0.0077822185373186414&0.0077822185373187065 \\
\hline
0.00012206286282867573&0.00012206286282875901\\
\hline
1.9073468138230965e-6&1.907346813826566e-6 \\
\hline
2.9802321943606103e-8&2.9802321943606116e-8 \\
\hline
4.656612873077393e-10&4.6566128719931904e-10 \\
\hline
7.275957614183426e-12&7.275957614156956e-12\\
\hline
1.1368683772161603e-13&1.1368683772160957e-13 \\
\hline
1.7763568394002505e-15&1.7763568394002489e-15 \\
\hline
0.0&2.7755575615628914e-17\\
\hline
\end{tabular}

Oczywiście bardziej zaufanym sposobem obliczania funkcji będzie $g(x)$ w której
błędy względne kolejnych zaokrągleń nie są aż tak wielkie jak również nie występuje
niebezpieczna operacja odejmowania liczb bliskich siebie. Jest to błąd
programistyczny którego należy być świadomym oraz należy unikać tego typu
niebezpiecznych operacji mogących wypaczyć wynik.
\section{zad. 7}

Zadanie polega na obliczeniu wartości pochodnej w punkcie dla $f(x)=sinx+cos3x$
$f'(x)=cosx-3sin3x$ w punkcie $x_0=1$
oraz obliczyć przybliżenie tego punktu za pomocą wzoru $f'(x_0)\approx
\overline{f}'(x_0)=\frac{f(x_0+h)-f(x_0)}{h}$ $h=2^{-n} (n=0,1,2,...,54)$

\begin{tabular}{|c|c|c|}
\hline
$\overline{f}'(x_0)$ & $f'(x)$ & $| f'(x_0)-\overline{f}'(x_0)|$\\
\hline
1.8704413979316472&-2.1149023979217905& 3.9853437958534377\\
\hline
1.1077870952342974&-1.0760038583593579& 2.1837909535936553\\
\hline
0.6232412792975817&-0.10661992002881371& 0.7298611993263954\\
\hline
0.3704000662035192&0.43883762041328944& 0.06843755420977027\\
\hline
0.24344307439754687&0.7186735647785978& 0.4752304903810509\\
\hline
0.18009756330732785&0.8593044249266039& 0.679206861619276\\
\hline
0.14849139537710958&0.9296634197019273& 0.7811720243308315\\
\hline
0.1327091142805159&0.9648369252723493& 0.8321278109918334\\
\hline
0.1248236929407085&0.9824200682346389& 0.8575963752939304\\
\hline
0.12088247681106168&0.991210473235729& 0.8703279964246673\\
\hline
0.11891225046883847&0.9956053511123194& 0.8766931006434809\\
```

```
\hline
0.11792723373901026&0.9978027047691286& 0.8798754710301183\\
\hline
0.11743474961076572&0.9989013597614758& 0.88146661015071\\
\hline
0.11718851362093119&0.9994506817341744& 0.8822621681132432\\
\hline
0.11706539714577957&0.9997253413315974& 0.8826599441858178\\
\hline
0.11700383928837255&0.9998626707820701& 0.8828588314936976\\
\hline
0.11697306045971345&0.9999313354201209& 0.8829582749604075\\
\hline
0.11695767106721178&0.9999656677173342& 0.8830079966501224\\
\hline
0.11694997636368498&0.9999828338604858& 0.8830328574968008\\
\hline
0.11694612901192158&0.9999914169306976& 0.883045287918776\\
\hline
0.1169442052487284&0.9999957084654625& 0.8830515032167341\\
\hline
0.11694324295967817&0.9999978542327597& 0.8830546112730815\\
\hline
0.11694276239722967&0.9999989271163869& 0.8830561647191573\\
\hline
0.11694252118468285&0.9999994635581952& 0.8830569423735124\\
\hline
0.116942398250103&0.9999997317790981& 0.8830573335289951\\
\hline
0.11694233864545822&0.9999998658895491& 0.8830575272440909\\
\hline
0.11694231629371643&0.9999999329447746& 0.8830576166510582\\
\hline
0.11694228649139404&0.9999999664723873& 0.8830576799809933\\
\hline
0.11694222688674927&0.9999999832361937& 0.8830577563494444\\
\hline
0.11694216728210449&0.9999999916180968& 0.8830578243359923\\
\hline
0.11694216728210449&0.9999999958090484& 0.8830578285269439\\
\hline
0.11694192886352539&0.9999999979045242& 0.8830580690409988\\
\hline
0.11694145202636719&0.9999999989522621& 0.8830585469258949\\
\hline
0.11694145202636719&0.999999999476131& 0.8830585474497639\\
\hline
0.11693954467773438&0.9999999997380655& 0.8830604550603312\\
\hline
0.116943359375&0.9999999998690328& 0.8830566404940328\\
\hline
0.1169281005859375&0.9999999999345164& 0.8830718993485789\\
\hline
0.116943359375&0.9999999999672582& 0.8830566405922582\\
\hline
0.11688232421875&0.9999999999836291& 0.8831176757648791\\
\hline
0.1168212890625&0.9999999999918145& 0.8831787109293145\\
\hline
0.116943359375&0.9999999999959073& 0.8830566406209073\\
\hline
0.11669921875&0.9999999999979536& 0.8833007812479536\\
```

```
\hline
0.1162109375&0.9999999999989768& 0.8837890624989768\\
\hline
0.1171875&0.9999999999994884& 0.8828124999994884\\
\hline
0.11328125&0.9999999999997442& 0.8867187499997442\\
\hline
0.109375&0.9999999999998721& 0.8906249999998721\\
\hline
0.109375&0.999999999999936& 0.890624999999936\\
\hline
0.09375&0.999999999999968& 0.906249999999968\\
\hline
0.125&0.999999999999984& 0.874999999999984\\
\hline
0.0&0.999999999999992& 0.999999999999992\\
\hline
0.0&0.999999999999996& 0.999999999999996\\
\hline
-0.5&0.999999999999998& 1.499999999999998\\
\hline
0.0&0.999999999999999& 0.999999999999999\\
\hline
\end{tabular} \\

 Widać iż błąd bezwzględny dla kolejnych wartości h na samym początku jest bardzo
duży, lecz w miarę zwiększania $h$ błąd maleje aby znów zacząć rosnąć do podobnych
rozmiarów jak na samym początku. Zadanie jest źle uwarunkowane gdyż małe zmiany
znacząco wpływają na wynik.

\end{document}
```