Obliczenia naukowe
Lista 1
Illia Azler | 239537

zad.1

a) Macheps (eps)

	Float16	Float32	Float64
Macheps (Calculated):	0.000977	1.1920929e-7	2.220446049250313e-16
Macheps (Julia):	0.000977	1.1920929e-7	2.220446049250313e-16

Wyniki pokazują, że zarówno obliczona precyzja maszynowa za pomocą funkcji find_eps, jak i wartości zwracane przez funkcję eps dla typów danych Float16, Float32 i Float64 są zgodne. Dla każdego z tych typów, wartość precyzji maszynowej jest rzędu wielkości oczekiwanej dla tych typów, co oznacza, że implementacja funkcji find_eps jest dokładna i zwraca oczekiwane wyniki.

b) Eta

	Float16	Float32	Float64
ETA (Calculated):	6.0e-8	1.0e-45	5.0e-324
ETA (Julia):	6.0e-8	1.0e-45	5.0e-324

Wyniki pokazują, że obliczona wartość ety za pomocą funkcji find_eta jest zgodna z wartościami zwracanymi przez funkcję nextfloat w języku Julia dla odpowiednich typów zmiennoprzecinkowych. Dla typów Float16, Float32 i Float64 otrzymujemy odpowiednio wyniki 6.0e-8, 1.0e-45 i 5.0e-324. Wartości te są rzędu wielkości oczekiwanej dla tych typów danych i są zgodne z wartościami zwracanymi przez funkcję nextfloat.

c) Liczba (MAX)

	Float16	Float32	Float64
Max (Calculated):	6.55e4	3.4028235e38	1.7976931348623157e308
Max (Julia):	6.55e4	3.4028235e38	1.7976931348623157e308

Wyniki pokazują, że obliczone maksymalne wartości za pomocą funkcji find_max są zgodne z wartościami zwracanymi przez funkcję floatmax dla odpowiednich typów zmiennoprzecinkowych. Dla typów Float16, Float32 i Float64 otrzymujemy odpowiednio wyniki 6.55e4, 3.4028235e38 i 1.7976931348623157e308. Te wartości są zgodne z oczekiwanymi wartościami maksymalnymi dla tych typów danych w języku Julia.

d) Odpowiedzi na pytania:

Jaki związek ma liczba macheps z precyzją arytmetyki?
Związek między liczbą macheps a precyzją arytmetyki polega na tym, że liczba macheps reprezentuje najmniejszą zmianę, którą można zobaczyć w danym typie zmiennej, podczas gdy precyzja arytmetyki odnosi się do ogólnej dokładności obliczeń.

Jaki związek ma liczba eta z liczbą MIN_{sub}?
Liczba eta ma związek z liczbą MIN_{sub} poprzez reprezentację minimalnej wartości, która może zostać odjęta od 1.0 w danym typie zmiennej. Obie te liczby odnoszą się do granic reprezentacji wartości w arytmetyce zmiennopozycyjnej.

Co zwracają funkcje floatmin(Float32) i floatmin(Float64) i jaki jest związek zwracanych wartości z liczbą MIN_{nor}?
Funkcje `floatmin(Float32)` i `floatmin(Float64)` zwracają najmniejsze dodatnie liczby możliwe do reprezentacji w danym typie zmiennopozycyjnym. MIN_{nor} odnosi się do minimalnej normalnej wartości, która może być reprezentowana w danym typie.

zad.2

$$3 \times (\frac{4}{3} - 1) - 1$$

	Float16	Float32	Float64
Kahan Epsilon:	-0.000977	1.1920929e-7	-2.220446049250313e-16
eps (Julia):	-0.000977	1.1920929e-7	2.220446049250313e-16

Porównanie wyników dla różnych arytmetyk z wartościami zwracanymi przez funkcję **eps** okazały się poprawne co do wartości bezwzględnej.

zad.3
Wyniki te są reprezentacją liczb zmiennoprzecinkowych w postaci binarnej za pomocą funkcji bitstring. Oto interpretacja poszczególnych wartości:

L.P.	Przedział [1;2] Liczba wchodząca [1.0000000000000002] $\delta = 2^{-52}$	Przedział [0.5;1] Liczba wchodząca [0.5000000000000001] $\delta = 2^{-53}$	Przedział [2;4] Liczba wchodząca [2.0000000000000004] $\delta = 2^{-51}$
1	0011111111100000000000000000000000000000000000000000000000000001	0011111111000000000000000000000000000000000000000000000000000001	0100000000000000000000000000000000000000000000000000000000000001
2	0011111111100000000000000000000000000000000000000000000000000010	0011111111000000000000000000000000000000000000000000000000000010	0100000000000000000000000000000000000000000000000000000000000010

3	0011111111100000000000000000000000000000000000000000000011	00111111111000000000000000000000000000000000000000000011	01000000000000000000000000000000000000000000000000000011
4	0011111111100000000000000000000000000000000000000000000100	00111111111000000000000000000000000000000000000000000100	01000000000000000000000000000000000000000000000000000100
5	0011111111100000000000000000000000000000000000000000000101	00111111111000000000000000000000000000000000000000000101	01000000000000000000000000000000000000000000000000000101
6	0011111111100000000000000000000000000000000000000000000110	00111111111000000000000000000000000000000000000000000110	01000000000000000000000000000000000000000000000000000110
7	0011111111100000000000000000000000000000000000000000000111	00111111111000000000000000000000000000000000000000000111	01000000000000000000000000000000000000000000000000000111
8	0011111111100000000000000000000000000000000000000000001000	00111111111000000000000000000000000000000000000000001000	01000000000000000000000000000000000000000000000000001000
9	0011111111100000000000000000000000000000000000000000001001	00111111111000000000000000000000000000000000000000001001	01000000000000000000000000000000000000000000000000001001
10	0011111111100000000000000000000000000000000000000000001010	00111111111000000000000000000000000000000000000000001010	01000000000000000000000000000000000000000000000000001010

Te wartości binarne reprezentują kolejne liczby zmiennoprzecinkowe w przedziale [1, 2] z krokiem $\delta = 2^{-52}$. Widzimy, że wartość mianownika rośnie o 1 dla każdego kolejnego kroku w tym przedziale, co potwierdza równomierne rozmieszczenie liczb zmiennoprzecinkowych.

W przedziale [0.5, 1] liczby zmiennoprzecinkowe są rozmieszczone podobnie jak w przedziale [1, 2] z tą różnicą, że krok (δ) jest mniejszy: $\delta = 2^{-53}$. To oznacza, że reprezentacje binarne kolejnych liczb będą różniły się na mniej znaczących bitach. W przedziale [2, 4] kroki będą większe, co oznacza, że kolejne reprezentacje binarne będą się różniły na bardziej znaczących bitach.

Dla przedziału [0.5, 1] liczby zmiennoprzecinkowe mogą być przedstawione za pomocą funkcji bitstring, która konwertuje je na ich reprezentację binarną. Odpowiednie iteracje z odpowiednim krokiem (δ) i początkową wartością będą generować reprezentacje kolejnych liczb w tym przedziale. Analogicznie dla przedziału [2, 4], reprezentacje będą generowane przy większym kroku ($\delta = 2^{-51}$).

zad.4

Program ten iteruje przez wartości zmiennopozycyjne w przedziale $1 < x < 2$ z krokiem 0.0001, sprawdzając warunek $x * (1/x) \neq 1$. Gdy znajdzie liczbę spełniającą ten warunek, wyświetla ją na ekranie.

Wynik programu:

```
Znaleziono liczbę x: 1.000000057228997
```

Wartość znaleziona jako przykład takiej liczby, która nie spełnia równości jest niewielka. Sugeruje to, że niektóre operacje arytmetyczne wykonywane na liczbach zmiennopozycyjnych mogą prowadzić do niewielkich błędów zaokrąglenia, które są niemal niewidoczne, ale wystarczające, aby zmienić wynik obliczeń na tyle, że nie spełnia on oczekiwanych warunków matematycznych.

zad.5

Program ten ilustruje problem utraty precyzji w obliczeniach zmiennopozycyjnych, szczególnie w przypadku sumowania dużych i małych liczb. W zależności od kolejności sumowania oraz znaków liczb, wynik może się różnić w zależności od kierunku sumowania, co prowadzi do różnic w precyzji dla różnych typów danych zmiennopozycyjnych.

Wyniki uzyskane dla typów danych Float32 i Float64 różnią się od prawidłowej wartości ($1.00657107000000 \cdot 10^{-11}$), co pokazuje ograniczenia precyzji w arytmetyce zmiennopozycyjnej, szczególnie w przypadku obliczeń na dużych zestawach danych. Wniosek naukowy z tego programu sugeruje ostrożność podczas wykonywania operacji sumowania w zmiennopozycyjnej arytmetyce i konieczność zrozumienia wpływu kolejności sumowania na wynik.

	Przód $\sum_{i=1}^{n} x_i y_i$	Tył $\sum_{i=n}^{1} x_i y_i$	DESC	ASC
Float32	-0.4999443	-0.4543457	-0,5	-0,5
Float64	1.025188136829667 2e-10	-1.5643308870494366e-10	0	0

zad.6

$$f(x) = \sqrt{x^2+1} - 1$$
$$g(x) = x^2/(\sqrt{x^2+1}+1)$$

x	f(x)	g(x)
8^{-1}	0.0077822185373186414	0.0077822185373187065
8^{-2}	0.00012206286282867573	0.00012206286282875901
8^{-3}	1.9073468138230965e-6	1.907346813826566e-6
8^{-4}	2.9802321943606103e-8	2.9802321943606116e-8
8^{-5}	4.656612873077393e-10	4.6566128719931904e-10
8^{-6}	7.275957614183426e-12	7.275957614156956e-12
8^{-7}	1.1368683772161603e-13	1.1368683772160957e-13
8^{-8}	1.7763568394002505e-15	1.7763568394002489e-15
8^{-9}	0.0	2.7755575615628914e-17
8^{-10}	0.0	4.336808689942018e-19

Wyniki pokazują, że funkcje f(x) i g(x) zwracają prawie identyczne wyniki dla większych wartości x, natomiast różnice zaczynają się stawać bardziej zauważalne dla bardzo małych wartości x.

Wartości dla f(x) oraz g(x) maleją wraz ze zmniejszaniem się x, aż osiągają wartość 0 dla x równego 8^{-9} oraz 8^{-10}. Różnice w wynikach są spowodowane różnicami w kolejności wykonywania działań arytmetycznych i ograniczeniami precyzji maszynowej.

Można stwierdzić, że w przypadku bardzo małych wartości x, wyniki obu funkcji są niewiarygodne, ponieważ komputer traci precyzję przy wykonywaniu operacji na tak małych liczbach. Jednak dla większych wartości x, wyniki są wiarygodne, ponieważ różnice są minimalne i wynikają głównie z różnic w kolejności wykonywania operacji arytmetycznych.

zad.7

h	$\tilde{f}$	$\lvert\tilde{f}(1) - f'(1)\rvert$	1+h	$\tilde{f}$ (1+h)	$\lvert f'(x_0) - \widetilde{f'}(x_0)\rvert$
2^{0}	2.0179892252685967	1.9010469435800585	2.0	1.8694677134760478	2.0179892252685967
2^{-1}	1.8704413979316472	1.753499116243109	1.5	0.7866991871732747	0.9352206989658236
2^{-2}	1.1077870952342974	0.9908448135457593	1.25	0.12842526201602544	0.27694677380857435
2^{-3}	0.6232412792975817	0.5062989976090435	1.125	-0.0706163518803512	0.07790515991219771
2^{-4}	0.3704000662035192	0.253457784514981	1.0625	-0.12537150765482896	0.02315000413771995
2^{-5}	0.2434430743975468 7	0.1265007927090087	1.03125	-0.14091391571762557	0.0076075960749233396
…	….	…..	…..	….	…..
2^{-25}	0.1169423982500103	1.1656156484463054e-7	1.0000000298023224	-0.14852150830739386	3.4851550534398257e-9
2^{-26}	0.1169423386454 5822	5.6956920069239914e-8	1.0000000149011612	-0.14852151004997227	1.7425766385414931e-9
2^{-27}	0.1169423162937 1643	3.460517827846843e-8	1.0000000074505806	-0.14852151092126076	8.712881527372929e-10
2^{-28}	0.1169422864913 9404	4.802855890773117e-9	1.0000000037252903	-0.14852151135690494	4.35643965346344e-10
2^{-29}	0.1169422268867 4927	5.4801788884617 51e-8	1.0000000018626451	-0.14852151157472704	2.1782187165086953e-10
2^{-30}	0.1169421672821 0449	1.1440643366000813e-7	1.0000000009313226	-0.14852151168363803	1.0891088031428353e-10
…	….	…..	…..	……	…….
2^{-44}	0.1171875	0.0002452183114618478	1.0000000000000568	-0.14852151179254225	6.661338147750939e-15
2^{-45}	0.11328125	0.0036610316885381 52	1.0000000000000284	-0.1485215117925457	3.219646771412954e-15
2^{-46}	0.109375	0.0075672816885381 52	1.0000000000000142	-0.14852151179254736	1.5543122344752192e-15
2^{-47}	0.109375	0.0075672816885381 52	1.000000000000007	-0.14852151179254813	7.771561172376096e-16
2^{-48}	0.09375	0.0231922816885381 52	1.0000000000000036	-0.14852151179254858	3.3306690738754696e-16

2^{-49}	0.125	0.008057718311461848	1.0000000000000018	-0.1485215117925487	2.220446049250313e-16
2^{-50}	0.0	0.11694228168853815	1.0000000000000009	-0.1485215117925489	0.0
2^{-51}	0.0	0.11694228168853815	1.0000000000000004	-0.1485215117925489	0.0
2^{-52}	-0.5	0.6169422816885382	1.0000000000000002	-0.14852151179254902	-1.1102230246251565e-16
2^{-53}	0.0	0.11694228168853815	1.0	-0.1485215117925489	0.0
2^{-54}	0.0	0.11694228168853815	1.0	-0.1485215117925489	0.0

Wnioskiem z wyników programu jest to, że w przypadku obliczeń różniczkowych numerycznych istnieje optymalna wartość h, po przekroczeniu której dalsze zmniejszanie h przestaje poprawiać przybliżenie i może nawet prowadzić do pogorszenia wyników. W początkowych etapach malejące h prowadzi do poprawy przybliżenia wartości pochodnej, ale tylko do pewnego momentu. Najlepsze przybliżenie uzyskujemy dla h = 2^{-28}.

Ponadto, analiza zachowania wartości 1+h sugeruje, że istnieje bezpośredni związek między tymi wartościami a poprawnością obliczeń różniczkowych. Początkowo zmniejszanie h prowadzi do zmniejszania wartości 1+h, co z kolei prowadzi do lepszego przybliżenia wartości funkcji w 1+h. Jednakże, po osiągnięciu pewnej wartości h, dalsze zmniejszanie h nie przynosi już dodatkowych korzyści.

Z tych obserwacji wynika, że w praktyce przy doborze optymalnej wartości h należy brać pod uwagę zarówno dokładność wyników, jak i stabilność obliczeń. Odpowiedni balans między tymi dwoma czynnikami jest kluczowy dla uzyskania wiarygodnych wyników obliczeń różniczkowych numerycznych.