Arytmetyka komputerowa - zadanie 12

Dany jest wielomian Wilkinsona $W(x) = (x-1)(x-2)...(x-20)$. Przekształcić wielomian do postaci $W(x) = x^{20} + a_{19}x^{19} + ... + a_1 x + a_0$. Podać wyznaczone wartości współczynników a_i. Dla $x = 1$ oraz $x = 20$ dokładna wartość wielomianu wynosi 0. Obliczyć wartość wielomianu dla $x = 1$ oraz $x = 20$. Do wyznaczenia wartości wielomianu wykorzystać schemat Hornera oraz dowolny schemat sumacyjny. Obliczenia wykonać dla zmiennych typu *float, double, long double*.

Specyfikacja sprzętu i środowiska

- System: Debian Linux Parrot OS x64
- Procesor: AMD Ryzen 5 4500U, 6 rdzeni, 6 wątków, 4.00GHz
- Pamięć RAM: 16 GB
- Język: C++, kompilator: gcc

Aga Patro, grupa 2

Kod wyznaczający współczynniki wielomianu

Tutaj jest ukazany kod dla typu float, jednak do operowania na innych typach użyłam kodu który zamiast na typie “float” to operuje na typie “double” oraz “long double”

```cpp
std::vector<float> multiply(std::vector<float>& A, std::vector<float>& B) {
    int m = A.size();
    int n = B.size();
    std::vector<float> prod( n: m + n - 1,  value: 0);

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            prod[i + j] += A[i] * B[j];
        }
    }
    return prod;
}

std::vector<float> multiply_wilkilson(int poly_number) {
    std::vector<float> first_poly = {-1.0f, 1.0f};
    std::vector<float> second_poly = {-2.0f, 1.0f};
    std::vector<float> prod = multiply( &: first_poly,  &: second_poly);

    if (poly_number == 2) {
        return prod;
    }

    for (int i = 3; i <= poly_number; i++) {
        std::vector<float> temp_poly = {static_cast<float>(-i), 1.0f};
        std::vector<float> new_prod = multiply( &: prod,  &: temp_poly);
        prod = new_prod;
    }
    return prod;
}
```

Wartości współczynników wielomianu w zależności od użytego typu

	float	double	long double
a0	2.4329e+18	2.4329e+18	2.4329e+18
a1	-8.75295e+18	-8.75295e+18	-8.75295e+18
a2	1.38038e+19	1.38038e+19	1.38038e+19
a3	-1.28709e+19	-1.28709e+19	-1.28709e+19
a4	8.03781e+18	8.03781e+18	8.03781e+18
a5	-3.59998e+18	-3.59998e+18	-3.59998e+18
a6	1.20665e+18	1.20665e+18	1.20665e+18
a7	-3.11334e+17	-3.11334e+17	-3.11334e+17
a8	6.30308e+16	6.30308e+16	6.30308e+16
a9	-1.01423e+16	-1.01423e+16	-1.01423e+16
a10	1.30754e+15	1.30754e+15	1.30754e+15

	float	double	long double
a11	-1.35585e+14	-1.35585e+14	-1.35585e+14
a12	1.13103e+13	1.13103e+13	1.13103e+13
a13	-7.56111e+11	-7.56111e+11	-7.56111e+11
a14	4.01718e+10	4.01718e+10	4.01718e+10
a15	-1.67228e+09	-1.67228e+09	-1.67228e+09
a16	5.33279e+07	5.33279e+07	5.33279e+07
a17	-1.25685e+06	-1.25685e+06	-1.25685e+06
a18	20615	20615	20615
a19	-210	-210	-210
a20	1	1	1

Jak widać współczynniki nie różnią się od siebie

Kod obliczający wartość wielomianu dla danego x schematem Hornera

```cpp
float horner(std::vector<float>& poly, float x) {
    float result = poly[poly.size()-1];

    for (int i = poly.size()-2; i >= 0; i--) {
        result = result * x + poly[i];
    }

    return result;
}
```

Tutaj jest ukazany kod dla typu float, jednak do operowania na innych typach użyłam kodu który zamiast na typie “float” to operuje na typie “double” oraz “long double”

Kod obliczający wartość wielomianu dla danego x schematem Clenshawa

```cpp
float clenshaw(std::vector<float>& poly, float x) {
    int n = poly.size() - 1;
    float b_n = poly[n];
    float b_n_minus_1 = poly[n - 1] + b_n * x;

    for (int i = n - 2; i >= 0; i--) {
        float temp = poly[i] + b_n_minus_1 * 2 * x - b_n;
        b_n = b_n_minus_1;
        b_n_minus_1 = temp;
    }

    return b_n_minus_1;
}
```

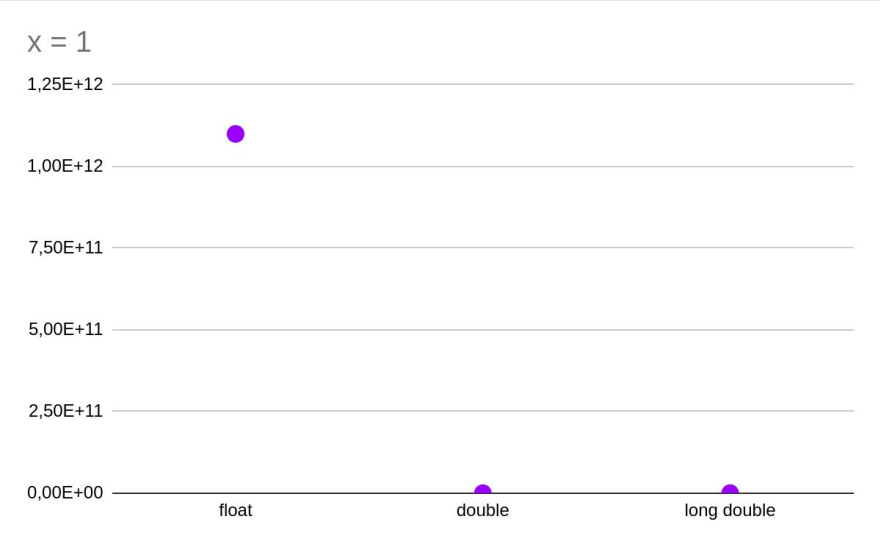
Wartości wielomianu wyliczone schematem Hornera dla x = 1 oraz x = 20, w zależności od typu użytej zmiennej

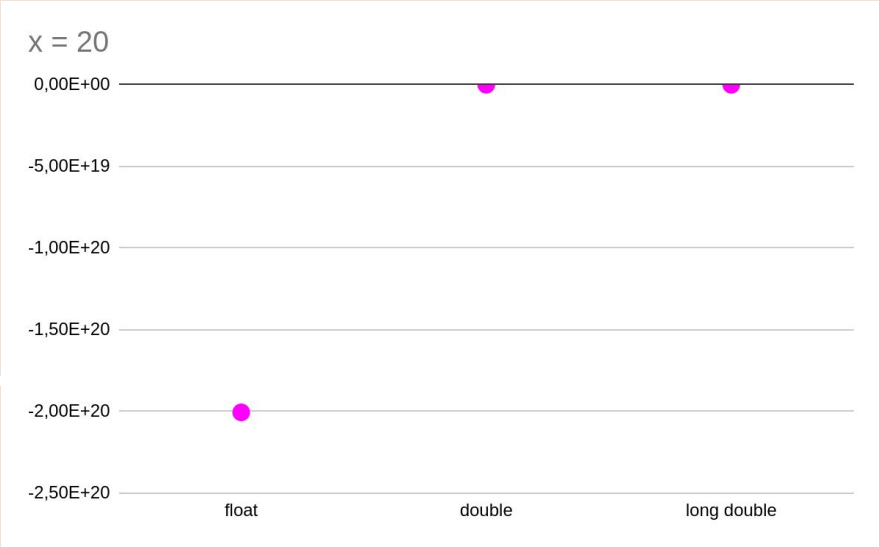
	float	double	long double
x = 1	1.09951e+12	1024	0
x = 20	-2.00515e+20	-2.70295e+10	0

Wartości wielomianu wyliczone schematem Clenshawa dla x = 1 oraz x = 20, w zależności od typu użytej zmiennej

	float	double	long double
x = 1	-1.21638e+17	-1.21645e+17	-1.21645e+17
x = 20	-5.41952e+31	-5.41952e+31	-5.41952e+31

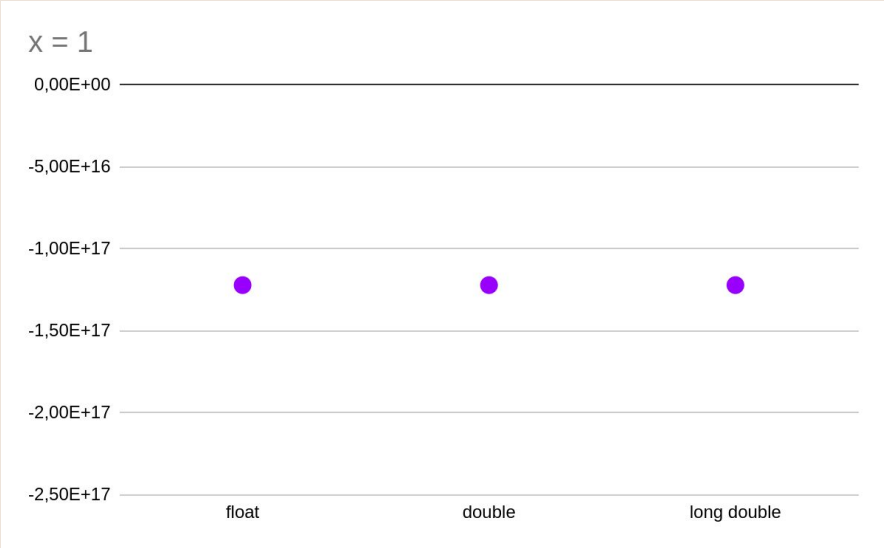
Porównanie wartości wielomianu wyliczone schematem Hornera dla różnych x, w zależności od typu użytej zmiennej

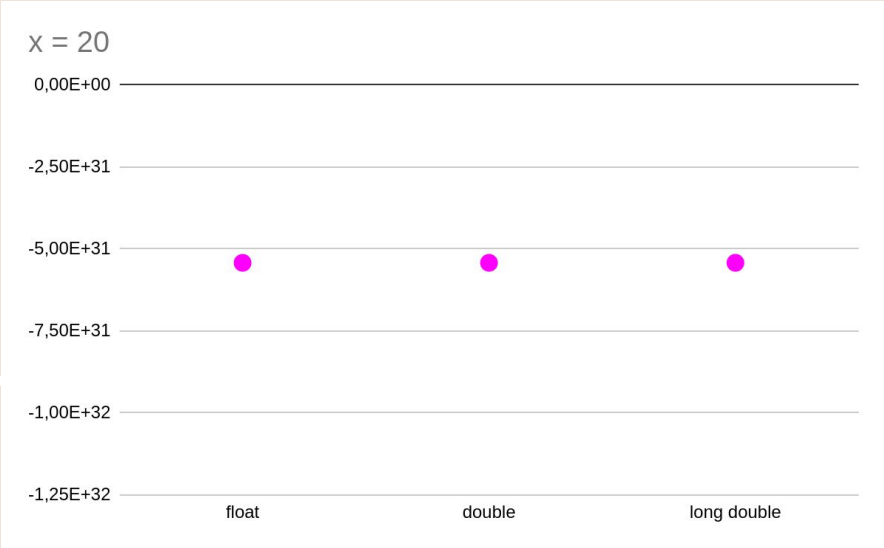




Zarówno dla x = 1 oraz x = 20 najbardziej zbliżone wyniki do oczekiwanych otrzymujemy dla typu “double” i “long double”

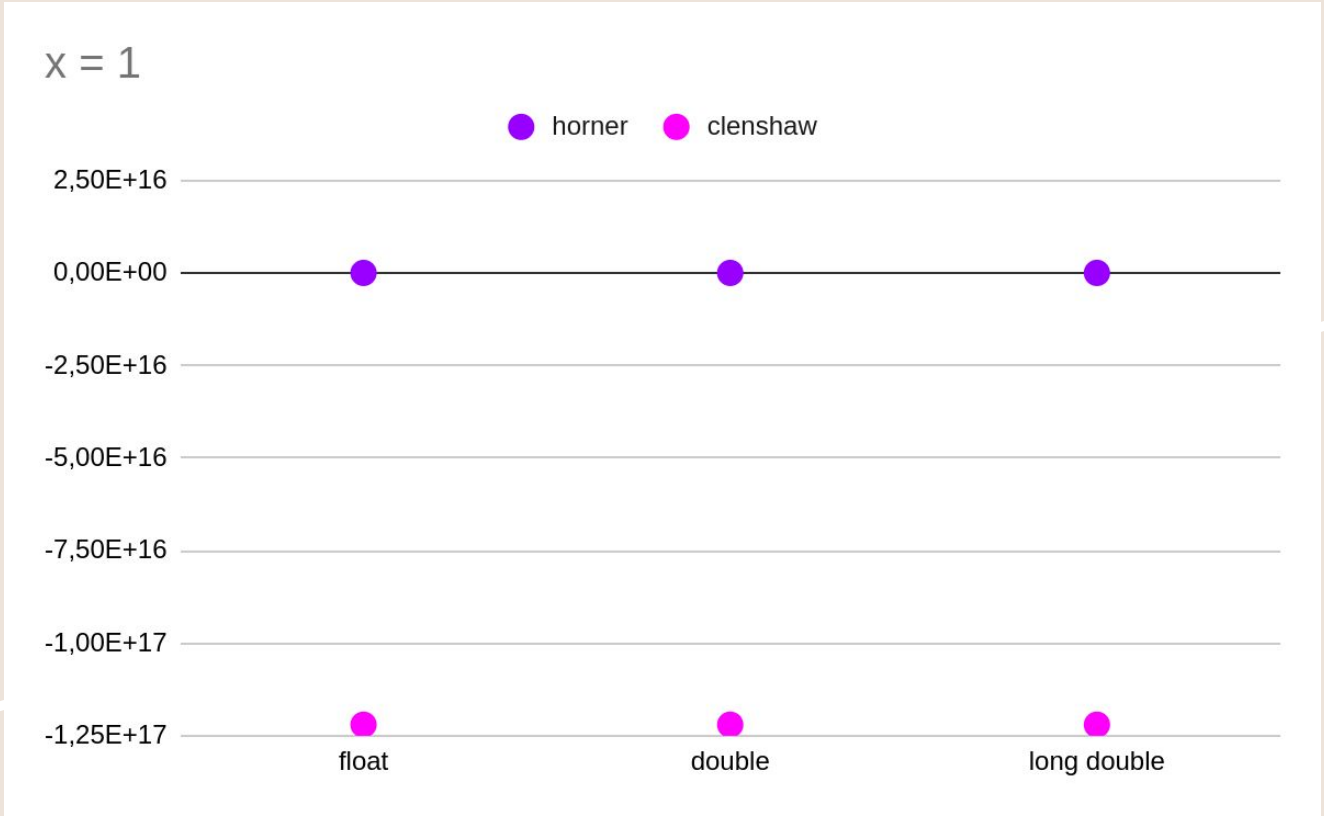
Porównanie wartości wielomianu wyliczone schematem Clenshawa dla różnych x, w zależności od typu użytej zmiennej





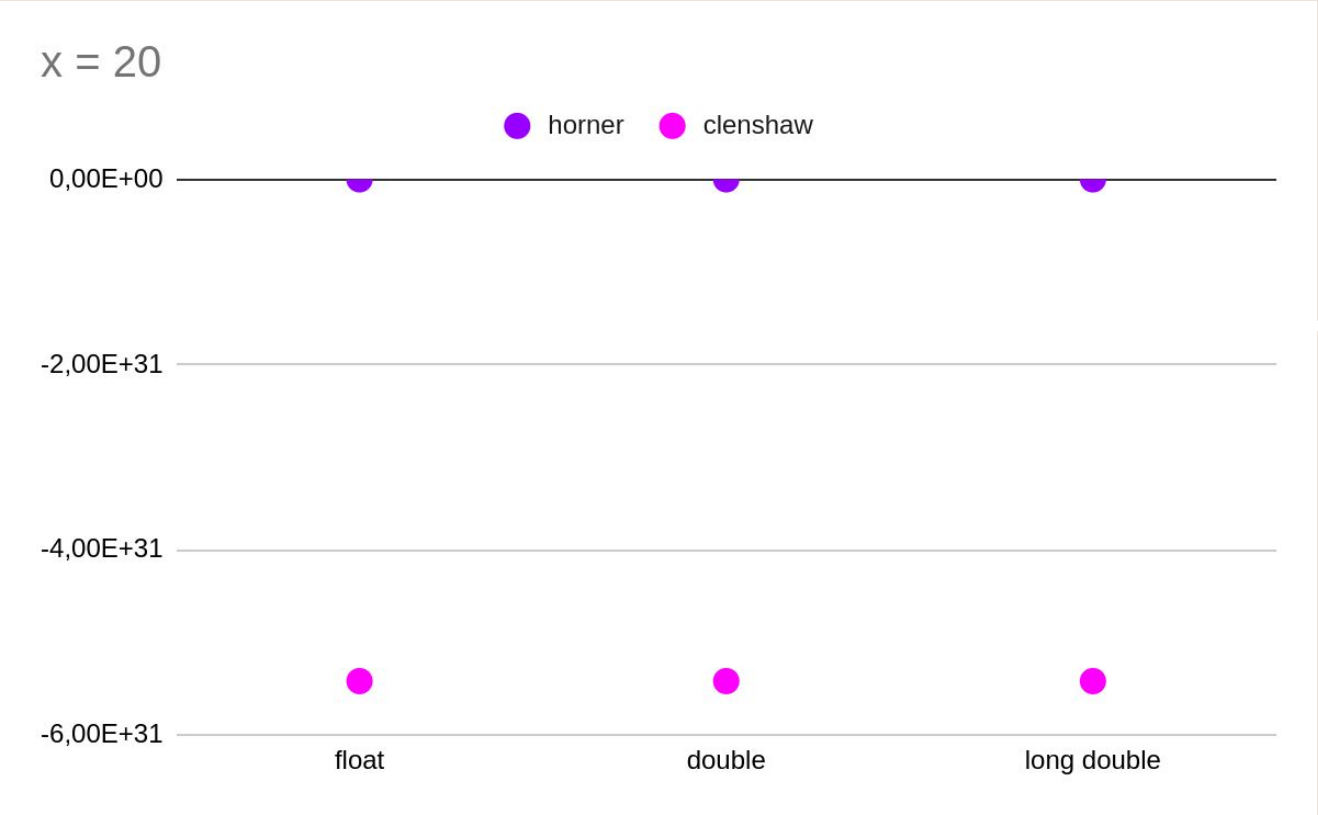
Zarówno dla x = 1 oraz x = 20 otrzymane wyniki nie różnią się od siebie w zależności od typu

Porównanie wartości wielomianu wyliczonych różnymi schematami w zależności od typu użytej zmiennej dla x = 1



Zgodnie z oczekiwaniami wyniki otrzymane dzięki schematowi Hornera są bardziej przybliżone do oczekiwanych.

Porównanie wartości wielomianu wyliczonych różnymi schematami w zależności od typu użytej zmiennej dla x = 20



Tak jak dla x = 1, zgodnie z oczekiwaniami wyniki otrzymane dzięki schematowi Hornera są bardziej przybliżone do oczekiwanych.

Wnioski

- Schemat Hornera i schemat Clenshawa są oba skutecznymi sposobami obliczania wartości wielomianu. Jednakże, dla konkretnych wartości x i stopnia wielomianu, jeden z nich może być dokładniejszy od drugiego.

- Typ danych ma wpływ na dokładność obliczeń. Wartości typu long double są bardziej dokładne niż wartości typu double, a te są bardziej dokładne niż wartości typu float.

- Wyniki dla schematu Clenshawa były takie same niezależnie od typu danych, co może wskazywać na to, że schemat ten jest bardziej stabilny niż schemat Hornera.

- Dla dokładnych obliczeń, zaleca się korzystanie z schematu Hornera lub Clenshawa w zależności od konkretnych warunków obliczeniowych, takich jak wartość x i stopień wielomianu.