Grzegorz Norbert Rogoziński, grupa 1I5

Projekt 1, zadanie 2.

Treść zadania:

Obliczyć wartość funkcji y=cos(x) korzystając z rozwinięcia:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

Dla różnych wartości x i dokładności ε. Zbadać zbieżność rozwinięcia (należy wykorzystać bibliotekę matematyczną języka C).

Program powinien posiadać menu oraz umożliwiać użytkownikowi kolejne obliczenia bez ponownego uruchamiania aplikacji (program ma działać, dopóki użytkownik go nie zakończy).

Sposób działania programu:

Program oblicza i sumuje kolejne wyrazy szeregu geometrycznego w postaci: $a_1 = 1$, $a_n = \frac{-a_{n-1} * x^2}{(2n-3)(2n-4)}$ (celem uniknięcia każdorazowego obliczania silni oraz kolejnych potęg być może dużych x), tak długo, aż znajdzie wyraz o module mniejszym od wprowadzonego przez użytkownika ε. Następnie wypisuje wynik oraz oblicza wartość funkcji ponownie, tym razem przyjmując dokładność ε/100. Jeśli oba wyniki są jednakowe, program informuje o najdokładniejszym możliwym do osiągnięcia wyniku, badając w ten sposób zbieżność funkcji.

Wczytywanie danych:

Podczas wczytywania wartości x i ε program sprawdza, czy są one liczbami i czy mieszczą się w możliwym do obliczenia zakresie. Zmienna x musi mieścić się między 10^{-9} a 10^9 (w czasie testów dla wartości odpowiednio mniejszych lub większych program zawieszał się). Dodatkowo wszystkie wartości spoza przedziału $[-20,20]$ są redukowane o wielokrotność 2π, aby początkowe wyrazy ciągu nie przyjmowały zbyt dużych wartości.

Wartość ε, dla większej wygody, wczytywana jest w postaci wykładnika potęgi liczby $\frac{1}{10}$ (dzięki temu niemożliwe jest m. in. wybranie dokładności równej 0). Ponadto musi mieścić się on w zakresie $[0, 10^9]$. Przy wykładniku ujemnym ε byłby większy od 1, co skutkowałoby bezsensownym wynikiem obliczeń, natomiast górna granica jest ustalona tak, jak przy x. Mimo tak dużego zakresu, poniżej $\varepsilon = 10^{-15}$obliczenia nie stają się już bardziej dokładne (o czym program informuje przed wprowadzaniem wartości).

Wartości obu zmiennych są przyjmowane tak długo, aż zostaną wprowadzone poprawne dane.

Badanie zbieżności:

Program bada zbieżność ciągu przez porównanie wyniku obliczeń z wynikiem dla stukrotnie mniejszego ε. Początkowo obiektem porównań miała być jakaś „wygodna” do przedstawienia wartość funkcji cosinus (np. $\cos\left(\frac{\pi}{3}\right) = 0.5$), jednak nie udawało mi się osiągnąć odpowiedniej dokładności obliczeń (bez względu na wprowadzany ε różnica między obliczaną a dokładną wartością wynosiła ok. 10^{-16}). Z tego powodu przyjąłem, że suma ciągu jest zbieżna do jej najdokładniejszej możliwej do obliczenia wartości, co nie jest do końca zgodne z prawdą.

Wprawdzie obecny sposób badania zbieżności wymusza wykonywanie tych samych obliczeń po raz drugi, co nie jest szczególnie optymalnym rozwiązaniem, jednak przy opisanych wcześniej ograniczeniach i sposobach uproszczenia podanego argumentu obliczenia te nie są bardzo długie ani wymagające.

Zapętlenie programu:

Po obliczeniu wartości funkcji program pyta użytkownika, czy ten chce ponownie wykonać jakieś obliczenia. Tak długo, jak wczytana zmienna char zawiera wartość ‘t’ lub ‘T’, niemal całość programu (poza deklaracjami zmiennych, wypisaniem informacji o autorze i przeznaczeniu aplikacji) jest wykonywana ponownie w pętli „do while”.

Opis wykonywanych testów wraz z wprowadzanymi zmianami:

Program od początku działał zgodnie z obecnym schematem z kilkoma pewnymi wyjątkami. Największą zmianą w stosunku do wcześniejszych wersji jest rezygnacja z funkcji służącej do obliczania rekurencyjnego silni, która musiała być wywoływana przy każdym wyrazie ciągu. Zamiast tego wprowadziłem dodatkową zmienną przechowującą wyraz a_{n-1}, aby same wyrazy obliczać za pomocą rekurencji.

Najwięcej uwagi wymagało ustalenie, dla jakich wartości podawanych zmiennych funkcja podaje (nie)prawidłowy wynik oraz zastosowanie odpowiednich rozwiązań. Przede wszystkim zająłem się wartościami x, dla których wynik znacznie tracił na dokładności (ok. |x|>40) poprzez stworzenie funkcji redukującej argument o iloczyn 2π i część całkowitą ilorazu x i 2π.

Kolejnym problemem było podanie wartości nie mieszczących się w zakresie zmiennych double. Po przeprowadzeniu szeregu testów okazało się, że przy liczbach dłuższych niż dziesięciocyfrowe program zawieszał się po wczytaniu danych. Choć double powinno móc przechowywać znacznie większe wartości, zamiast zagłębiać się w przyczynę problemu ograniczyłem wprowadzane wartości do $\pm 10^9$, uznając że jest to satysfakcjonujący zakres argumentów.

Napotkałem również problem przy badaniu zbieżności, który opisany już był w akapitach powyżej.