Lista 8 – Wstęp do informatyki, W. Hebisch

Zadanie 1

Funkcja $f : [0,1] \mapsto [0,1]$ zadana jest wzorem: $f(x) = 4x(1-x)$. Dla zadanej liczby x_0 definiujemy ciąg $\{x_n\}$ przy pomocy zależności rekurencyjnej $x_{n+1} = f(x_n)$ (tzn. x_n jest n-krotną iteracją f na x_0. Napisać program który dla zadanych x_0 i y_0 obliczy kolejne wartości x_n i y_n. Dla bliskich x_0 i y_0 wypisać wartości i różnice między x_n i y_n dla $n = 0, 10, 20, ..., 100$.

Zadanie 2

Spróbować obliczyć przybliżoną wartość funkcji wykładniczej sumując szereg potęgowy dla wartości x równych $-20, -30, ..., -100$. Czy powiększanie liczby wyrazów pomaga?

Zadanie 3

Zaprogramować obliczanie wielomianu $W(x) = (x-1)^{10}$ sumując poszczególne wyrazy. Wypisać wartości W na przedziale $[0.98, 1.02]$ z krokiem co 0.001. Jak zachowuje znak $W(x)$? Porównać z wartościami obliczonymi bezpośrednio z definicji $W(x)$.

Zadanie 4

Doświadczalnie zbadać zachowanie rozwiązań równania kwadratowego w pobliżu pierwiastka podwójnego (np. prawą stronę równania $x^2 - 2x + 1 = 0$ zastąpić przez małą liczbę). Znaleźć przykład gdy niedokładność jest widoczna.

Zadanie 5

Zbadać doświadczalnie błędy rozwiązania układów równań (skorzystać z programu przykładowego gauss.c). Dobrać tak układ równań żeby błąd przekroczył 1% rozwiązania.

Zadanie 6

Zmodyfikować program gauss.c, tak żeby rozwiązywać na raz m układów równań z tą samą lewą stroną (zastąpić wektor y przez tablicę).

Zadanie 7

W programie odwracającym linie (rewlin.c) musimy obliczać długość lini mimo że funkcja `czytaj_linie` wie jaka długa była linia. Używając deklarację:

```
typedef struct { char * txt; long dlugosc;} linia;
```

zmodyfikować funkcję `czytaj_linie` tak by zwracana wartość była typu `linia` (i zawierała zarówno wskaźnik do bufora, jak i długość linii). Zrobić też odpowiednie zmiany do programu głównego (w szczególności usunąć wywołanie funkcji `strlen`).