Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1

з дисципліни
«Алгоритми і структури даних»

Виконав:
студент групи ІМ-31
Литвиненко Сергій Андрійович
номер у списку групи: 12

Перевірила:
Молчанова А. А.

Київ 2024

Завдання

Дане натуральне число n. Знайти суму перших n членів ряду чисел, заданого рекурентною формулою. Розв'язати задачу **трьома способами**:

1) у програмі використати рекурсивну функцію, яка виконує обчислення і членів ряду, і суми на рекурсивному спуску;

2) у програмі використати рекурсивну функцію, яка виконує обчислення і членів ряду, і суми на рекурсивному поверненні;

3) у програмі використати рекурсивну функцію, яка виконує обчислення членів ряду на рекурсивному спуску, а обчислення суми на рекурсивному поверненні.

При проєктуванні програм слід врахувати наступне:

1) програми повинні працювати коректно для довільного цілого додатного n включно з n = 1;

2) видимість змінних має обмежуватися тими ділянками, де вони потрібні;

3) функції повинні мати властивість модульності;

4) у кожному з трьох способів рекурсивна функція має бути одна (за потреби, можна також використати додаткову функцію-обгортку (wrapper function));

5) у другому способі можна використати запис (struct) з двома полями (але в інших способах у цьому немає потреби і це вважатиметься надлишковим);

6) програми мають бути написані мовою програмування С.

Варіант 12

$$F_1 = 1; \quad F_{i+1} = -F_i \cdot x^2/(4i^2 - 2i), \qquad i > 0;$$

Текст програм

1. Рекурсивну функція, яка виконує обчислення і членів ряду, і суми на рекурсивному спуску.

```
#include <stdio.h>


double wrapMyCos(double x, unsigned i, unsigned n, double previos, double
totalSum) {
  if (i >= n) return totalSum;
  const double next = -previos * (x * x) / (4 * i * i - 2 * i);
  const double sum = totalSum + next;
  return wrapMyCos(x, i + 1, n, next, sum);
}


double myCos(double x, unsigned n) {
  return wrapMyCos(x, 1, n, 1.0, 1.0);
}


int main(int argc, char const *argv[]) {
  const unsigned n = 5;
  double x;
  printf("Enter x: ");
  scanf("%lf", &x);
  const double res = myCos(x, n);
  printf("cos(%lf) = %lf\n", x, res);
  return 0;
}
```

2. Рекурсивна функція, яка виконує обчислення і членів ряду, і суми на рекурсивному поверненні.

```
#include <stdio.h>

double wrapMyCos(double x, unsigned n, double* previos) {
  if (n <= 1) return *previos;
  const double totalSum = wrapMyCos(x, n - 1, previos);
  const unsigned i = n - 1;
  const double next = -*previos * (x * x) / (4 * i * i - 2 * i);
  const double sum = totalSum + next;
  *previos = next;
  return sum;
}

double myCos(double x, unsigned n) {
  double previos = 1.0;
  return wrapMyCos(x, n, &previos);
}

int main(int argc, char const *argv[]) {
  const unsigned n = 5;
  double x;
  printf("Enter x: ");
  scanf("%lf", &x);
  const double res = myCos(x, n);
  printf("cos(%lf) = %lf\n", x, res);
  return 0;
}
```

3. У програмі використати рекурсивну функцію, яка виконує обчислення членів ряду на рекурсивному спуску, а обчислення суми на рекурсивному поверненні.

```
#include <stdio.h>

double wrapMyCos(double x, unsigned i, unsigned n, double previos) {
  if (i >= n) return 1.0;
  const double next = -previos * (x * x) / (4 * i * i - 2 * i);
  const double totalSum = wrapMyCos(x, i + 1, n, next);
  const double sum = next + totalSum;
  return sum;
}

double myCos(double x, unsigned n) {
  return wrapMyCos(x, 1, n, 1.0);
}

int main(int argc, char const *argv[]) {
  const unsigned n = 5;
  double x;
  printf("Enter x: ");
  scanf("%lf", &x);
  const double res = myCos(x, n);
  printf("cos(%lf) = %lf\n", x, res);
  return 0;
}
```

4. Циклічний варіант вирішення задачі.

```
#include <stdio.h>


double myCos(double x, unsigned n) {
  double res = 1.0;
  double previos = res;
  for (int i = 1; i < n; i++) {
    previos = -previos * (x * x) / (4 * i * i - 2 * i);
    res += previos;
  }
  return res;
}


int main(int argc, char const *argv[]) {
  const unsigned n = 5;
  double x;
  printf("Enter x: ");
  scanf("%lf", &x);
  const double res = myCos(x, n);
  printf("cos(%lf) = %lf\n", x, res);
  return 0;
}
```

Тестування програм

1. Рекурсивну функція, яка виконує обчислення і членів ряду, і суми на рекурсивному спуску.

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\descent.exe
Enter x: 3.141592
cos(3.141592) = -0.976022
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\descent.exe
Enter x: 1.570796
cos(1.570796) = 0.000025
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\descent.exe
Enter x: 1.047197
cos(1.047197) = 0.500001
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\descent.exe
Enter x: -3.141592
cos(-3.141592) = -0.976022
```

2. Рекурсивна функція, яка виконує обчислення і членів ряду, і суми на рекурсивному поверненні.

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\comeback.exe
Enter x: 3.141592
cos(3.141592) = -0.976022
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\comeback.exe
Enter x: 1.570796
cos(1.570796) = 0.000025
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\comeback.exe
Enter x: 1.047197
cos(1.047197) = 0.500001
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\comeback.exe
Enter x: -3.141592
cos(-3.141592) = -0.976022
```

3. Рекурсивна функція, яка виконує обчислення членів ряду на рекурсивному спуску, а обчислення суми на рекурсивному поверненні.

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\mixed.exe
Enter x: 3.141592
cos(3.141592) = -0.976022
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\mixed.exe
Enter x: 1.570796
cos(1.570796) = 0.000025
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\mixed.exe
Enter x: 1.047197
cos(1.047197) = 0.500001
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\mixed.exe
Enter x: -3.141592
cos(-3.141592) = -0.976022
```

4. Циклічний варіант вирішення задачі.

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\iteration.exe
Enter x: 3.141592
cos(3.141592) = -0.976022
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\iteration.exe
Enter x: 1.570796
cos(1.570796) = 0.000025
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\iteration.exe
Enter x: 1.047197
cos(1.047197) = 0.500001
```

```
PS D:\lessons\semester2\algorithms_and_data_structures\lab1> .\iteration.exe
Enter x: -3.141592
cos(-3.141592) = -0.976022
```

Перевірка на калькуляторі:

Щоб перевірити правильність обчислень елементів ряду, я вивів на екран відповідну величину та порівняв її з власними розрахунками.

1. Рекурсивну функція, яка виконує обчислення і членів ряду, і суми на рекурсивному спуску.

```
PS D:\lessons\ads\semester2\lab1> .\descent.exe
Enter x: 3.141592
F1 descent: 1.000000
F2 descent: -4.934800
F3 descent: 4.058709
F4 descent: -1.335261
F5 descent: 0.235330
cos(3.141592) = -0.976022
```

2. Рекурсивна функція, яка виконує обчислення і членів ряду, і суми на рекурсивному поверненні.

```
PS D:\lessons\ads\semester2\lab1> .\comeback.exe
Enter x: 3.141592
F1 comeback: 1.000000
F2 comeback: -4.934800
F3 comeback: 4.058709
F4 comeback: -1.335261
F5 comeback: 0.235330
cos(3.141592) = -0.976022
```

3. Рекурсивна функція, яка виконує обчислення членів ряду на рекурсивному спуску, а обчислення суми на рекурсивному поверненні.

```
PS D:\lessons\ads\semester2\lab1> .\mixed.exe
Enter x: 3.141592
F1 mixed: 1.000000
F2 mixed: -4.934800
F3 mixed: 4.058709
F4 mixed: -1.335261
F5 mixed: 0.235330
cos(3.141592) = -0.976022
```

4. Циклічний варіант вирішення задачі.

```
PS D:\lessons\ads\semester2\lab1> .\iteration.exe
Enter x: 3.141592
F1 iteration: 1.000000
F2 iteration: -4.934800
F3 iteration: 4.058709
F4 iteration: -1.335261
F5 iteration: 0.235330
cos(3.141592) = -0.976022
```

За власними розрахунками:

$$F_1 = 1$$

$$F_2 = -1 \cdot \frac{3.141592^2}{4 \cdot 1^2 - 2 \cdot 1} = -4.934800147$$

$$-1 \cdot \frac{3.141592^2}{4 \cdot 1^2 - 2 \cdot 1} = -4.934800147$$

$$F_3 = -(-4.934800147) \cdot \frac{3.141592^2}{4 \cdot 2^2 - 2 \cdot 2} = 4.058708749$$

$$-(-4.934800147) \cdot \frac{3.141592^2}{4 \cdot 2^2 - 2 \cdot 2} = 4.058708749$$

$$F_4 = -4.058708749 \cdot \frac{3.141592^2}{4 \cdot 3^2 - 2 \cdot 3} = -1.335261102$$

$$-4.058708749 \cdot \frac{3.141592^2}{4 \cdot 3^2 - 2 \cdot 3} = -1.335261102$$

$$F_5 = -(-1.335261102) \cdot \frac{3.141592^2}{4 \cdot 4^2 - 2 \cdot 4} = 0.2353302387$$

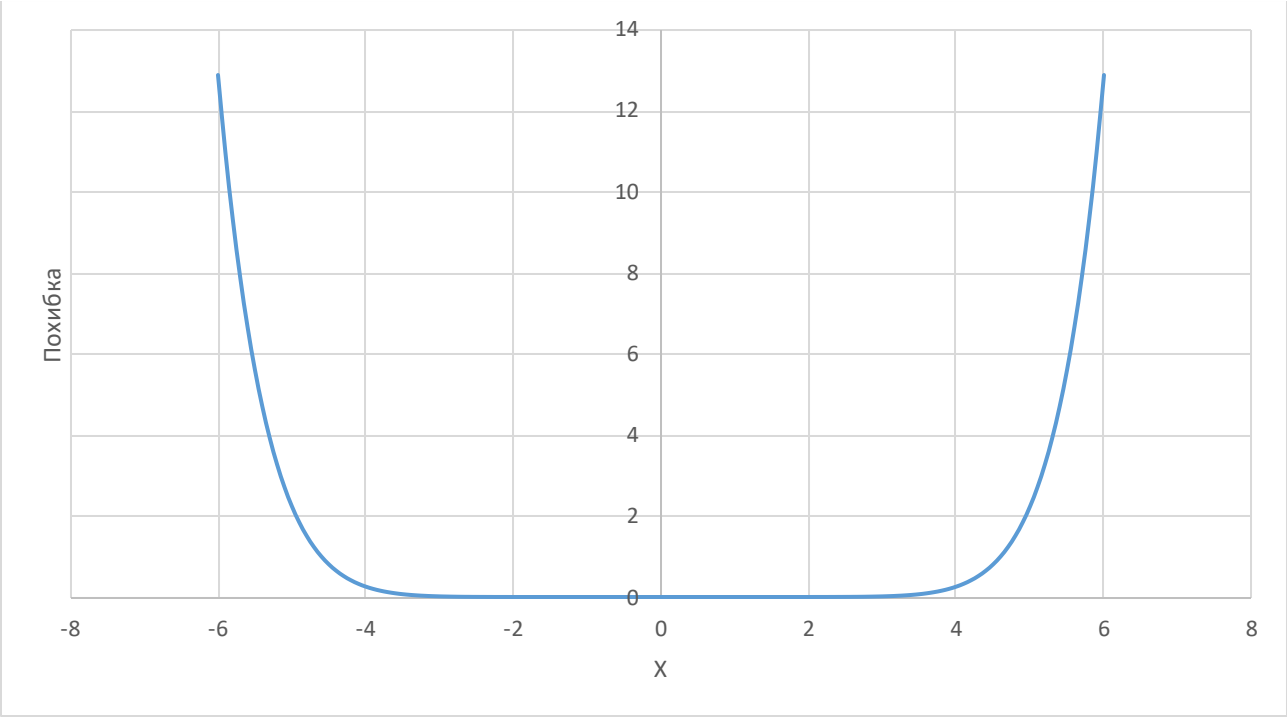
$$-(-1.335261102) \cdot \frac{3.141592^2}{4 \cdot 4^2 - 2 \cdot 4} = 0.2353302387$$

Сума також співпадає з обчисленнями програми.

$$1 - 4.934800147 + 4.058708749 + -1.335261102 + 0.2353302387 = -0.9760222613$$

Отже, програми працюють правильно.

Графік похибки обчислення функції



Висновок

Протягом виконання лабораторної роботи я навчився розробляти рекурсивні алгоритми. Було розроблено 4 програми для апроксимації функції cos. Точність обчислення функції за допомогою апроксимації залежить від кількості складених доданків, тож чим більший параметр n в функції, що апроксимує, тим більш точним є результат обчислення. Даний метод апроксимації розкладає функцію cos в околі нуля, тож чим ближче значення аргументу знаходиться до нуля, тим менше ітерацій або рекурсивних викликів знадобиться для обчислення точного значення функції. І навпаки, для того, щоб обчислити значення функції в точці, що знаходиться далеко від нуля, треба більша кількість ітерацій або рекурсивних викликів, що гарно видно на графіку, представленому вище.

Рекурсивні алгоритми інколи є більш наглядними та короткими, ніж їх ітеративний варіант, особливо коли задача поставлена рекурентним відношенням. Вони є незамінними в тих випадках, коли алгоритм майже неможливо представити ітераційним способом. Проте, рекурсивні алгоритми часто є повільнішими за їх ітеративний аналог, адже витрачаються додаткові ресурси для виклику функції та поверненню з неї. Також рекурсивні алгоритми вимагають більше пам’яті через зберігання контексту функції.