МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
«НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут телекомунікацій, радіоелектроніки та електронної техніки
Кафедра «Радіоелектронні пристрої та системи»

Звіт з лабораторної роботи № 5А

«Програмування, частина 2»

Підготував:
ст. гр. ІХ-11
Диркавець Максим
Перевірив:
Асистент каф РЕПС
Чайковський І.Б.

Львів 2024

Тема: Дослідження особливостей використання вказівників у мові С

Мета роботи: Дослідити властивості циклічних операторів мови С.

Хід роботи

1.Здійснити табулювання функції, що з певними припущеннями з достатньою точністю моделює імпульс Максвела, який утворюється при ударному збудженні широкосмугової антени. Обчислення провести на проміжку зміни і в межах [0-31] з кроком i=1, N=32. Результати вивести у вигляді таблиці. Визначити найбільше та найменше значення функції на цьому проміжку.

$$y = i^2 e^{-i^2/100} \sin(\frac{2\pi}{N} i)$$

```
int main() {
    int N = 32;
    double y, min_value, max_value;
    int min_i, max_i;
    min_value = INFINITY;
    max_value = -INFINITY;
    printf(" i  |         y\n");
    printf("----|--------------------\n");
    for (int i = 0; i <= 31; i++) {
        y = i * i * exp(-i * i / 100.0) * sin(2 * M_PI * i / N);
        printf("%2d  | %0.10f\n", i, y);
        if (y < min_value) {
            min_value = y;
            min_i = i;
        }
        if (y > max_value) {
            max_value = y;
            max_i = i;
```

```
        }
    }
    printf("\nНайменше значення функції: y(%d) = %0.10f\n", min_i, min_value);
    printf("Найбільше значення функції: y(%d) = %0.10f\n", max_i, max_value);
    return 0;
}
```

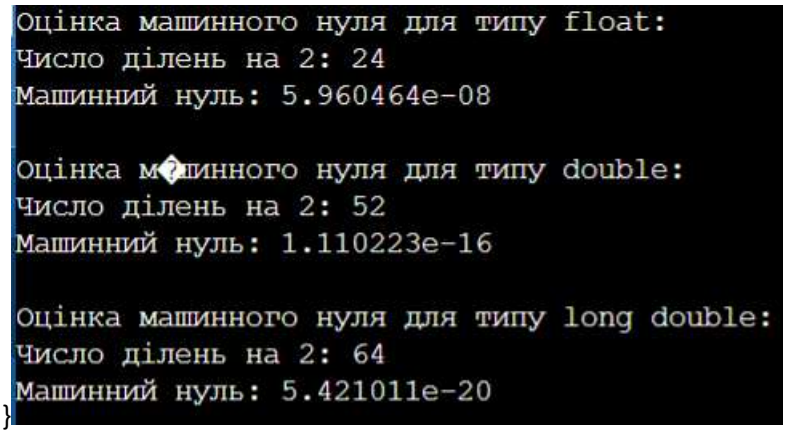
i	y
0	0.0000000000
1	0.1931491409
2	1.4707128014
3	4.5697766541
4	9.6409064276
5	16.1887296290
6	23.2044796071
7	29.4418925403
8	33.7467151388
9	35.3411297008
10	33.9876286130
11	30.0009791344
12	24.1247843734
13	17.3247907851
14	10.5652200631
15	4.6265329485
16	0.0000000000
17	-3.1334481711
18	-4.8559091116
19	-5.4255415130
20	-5.1804449840
21	-4.4570376833
22	-3.5357000540
23	-2.6158437683
24	-1.8150402807
25	-1.1833506258
26	-0.7239879605
27	-0.4135872452
28	-0.2182389818
29	-0.1040203646
30	-0.0425041987
31	-0.0125715591

```
Найменше значення функції: y(19) = -5.4255415130
Найбільше значення функції: y(9) = 35.3411297008
```

2. В обчислювальних задачах при програмуванні ітераційних алгоритмів, що закінчуються при досягненні заданої точності, часто необхідна оцінка «машинного нуля», тобто числового значення, менше за яке неможливо задати точність даного алгоритму. Абсолютне значення «машинного нуля» залежить від розрядної сітки застосовуваного комп'ютера, від прийнятої в конкретному трансляторі точності представлення дійсних чисел і від значень, що використовуються для оцінки точності. Наступна програма оцінює абсолютне значення «машинного нуля» відносно близьких (за модулем) до одиниці змінних типу float.

```
#include <stdio.h>
#include <math.h>

void evaluate_machine_epsilon_float();
void evaluate_machine_epsilon_double();
void evaluate_machine_epsilon_long_double();

int main(void) {
   printf("Оцінка машинного нуля для типу float:\n");
   evaluate_machine_epsilon_float();

   printf("\nОцінка машинного нуля для типу double:\n");
   evaluate_machine_epsilon_double();

   printf("\nОцінка машинного нуля для типу long double:\n");
   evaluate_machine_epsilon_long_double();

   return 0;
}

void evaluate_machine_epsilon_float() {
   float precision = 1.0f, a;
   int i = 0;

   while (1) {
      precision /= 2.0f;
      a = precision + 1.0f;
      i++;
      if (a <= 1.0f) break;
   }

   printf("Число ділень на 2: %d\n", i);
   printf("Машинний нуль: %e\n", precision);
}
```

```
void evaluate_machine_epsilon_double() {
   double precision = 1.0, a;
   int i = 0;

   for (i = 0;; i++) {
      precision /= 2.0;
      a = precision + 1.0;
      if (a <= 1.0) break;
   }

   printf("Число ділень на 2: %d\n", i);
   printf("Машинний нуль: %le\n", precision);
}

void evaluate_machine_epsilon_long_double() {
   long double precision = 1.0L, a;
   int i = 0;

   do {
      precision /= 2.0L;
      a = precision + 1.0L;
      i++;
   } while (a > 1.0L);


   printf("Число ділень на 2: %d\n", i);

   printf("Машинний нуль: %Le\n", precision);
```




```
}
```

3. Заповнити екран монітора символами так, щоб утворити прямокутний трикутник зображений на рисунку, використавши для цього вкладені цикли.

```
#include <stdio.h>

int main() {

   int height = 10;

   for (int i = 1; i <= height; i++) {
```

```
      for (int j = 1; j <= i; j++) {
         printf("*");
      }
      printf("\n");
   }
   return 0;
}
```

4. Обчислити значення скінченої суми, або добутку згідно свого варіанту. Врахувати, що навіть для невеликих чисел значення факторіала може вийти за гранично допустимі для даного типу даних. Аргумент тригонометричних функцій задавати в межах:

```
#include <stdio.h>
#include <math.h>
unsigned long long factorial(int n) {
    unsigned long long result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
int main() {
    int N;
    printf("Введіть натуральне число N: ");
```

```
    scanf("%d", &N);
    double S = 0.0;
    for (int k = 1; k <= N; k++) {
        unsigned long long fact = factorial(k);
        double ln_fact = log((double)fact);
        S += k * k * ln_fact;
    }
    printf("Сума S = %.10lf\n", S);
    return 0;
}
```

```
Введіть натуральне число N: 4
Сума S = 69.7472852309
```

5. Відомо, що одним із методів обчислення багатьох функцій є розкладання їх у ряд Тейлора:

Завдання: для заданого x, яке уводиться з клавіатури під час роботи програми, обчислити значення функції у за допомогою бібліотечних функцій компілятора так і за допомогою вище наведеного явного розкладу її в ряд (ітераційний процес до досягнення заданої точності). Обчислити при цьому також кількість ітерацій або кількість членів ряду в розкладі функції. Точність обчислень, тобто значення члена ряду розкладу функції коли необхідно припиняти ітераційний процес, a=0.00001. Аргумент тригонометричних функцій задавати в межах: 0 X

```
#include <stdio.h>
#include <math.h>
#define EPSILON 0.00001
#define PI 3.141592653589793
double taylor_sin(double x, int *iterations) {
    double term = x;
    double sum = term;
    int n = 1;
```

```
    while (fabs(term) >= EPSILON) {
        term *= -x * x / ((2 * n) * (2 * n + 1));
        sum += term;
        n++;
    }
    *iterations = n;
    return sum;
}
int main() {
    double x;
    printf("Введіть значення x (0 ≤ x ≤ π): ");
    scanf("%lf", &x);
    if (x < 0 || x > PI) {
        printf("Помилка: x повинно бути в межах [0, π]\n");
        return 1;
    }
    double library_sin = sin(x);
    int iterations;
    double taylor_sin_value = taylor_sin(x, &iterations);
    printf("sin(x) за допомогою бібліотечної функції: %.10lf\n", library_sin);
    printf("sin(x) за допомогою ряду Тейлора: %.10lf\n", taylor_sin_value);
    printf("Кількість ітерацій: %d\n", iterations);
    return 0;
}
```

```
Введіть значення x (0 ≤ x ≤ π): 3
sin(x) за допомогою бібліотечної функції: 0.1411200081
sin(x) за допомогою ряду Тейлора: 0.1411200174
Кількість ітерацій: 9
```

Висновок: У даному дослідженні розглянуто використання вказівників у мові програмування С та їх роль у розробці алгоритмів і моделей. Було досягнуто

наступних результатів:

1. **Табулювання функції**: Моделювання імпульсу Максвела проведено з заданими параметрами, результати табулювання виведено у вигляді таблиці. Визначено максимальне та мінімальне значення функції на проміжку від 0 до 31.

2. **Оцінка машинного нуля**: Розроблено програму для визначення абсолютного значення "машинного нуля" для типу даних **float**. Ця оцінка є критично важливою для забезпечення точності в ітераційних алгоритмах.

3. **Графічний вивід**: Реалізовано програму для заповнення екрану символами, що утворюють прямокутний трикутник. Це завдання підкреслює ефективність використання вкладених циклів у мові C.

4. **Обчислення скінченої суми або добутку**: Проведено обчислення з урахуванням можливого виходу факторіала за гранично допустимі значення для певного типу даних. Це дозволило проаналізувати обмеження, що накладаються типами даних у C.

5. **Розклад функцій у ряд Тейлора**: Для заданого аргументу x, введеного з клавіатури, обчислено значення функції як за допомогою бібліотечних функцій, так і шляхом розкладу у ряд Тейлора до заданої точності. Було визначено кількість ітерацій або членів ряду, необхідних для досягнення точності 0.00001.

Загалом, дослідження продемонструвало важливість правильного використання вказівників, вкладених циклів, та методів оцінки точності у мові C. Отримані результати та їх аналіз допоможуть у подальшій розробці ефективних алгоритмів і програм.