Universidade de São Paulo
Instituto de Física de São Carlos

Lista 1

Pedro Calligaris Delbem 5255417

Professor: Attilio Cucchieri

Março de 2025

Sumário

1 Exercício 1

Tarefa: Calcular a área de um círculo. Opções: entrada e saída usando teclado e monitor; entrada e saída usando arquivos; calcular a área em uma subrotina.

Código Escrito:

```
!-------------------------------------------------------------
! File: L1-5255417-ex-1.f90
!
! Description:
!   Computes the area of a circle.
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!-------------------------------------------------------------
program circle_area

    !deactivate implicit typing
    implicit none

    !define variables
    real radio, area

    !request radius value to the user
    write(*,*) 'Insert radius value:'

    !read user input
    read(*,*) radio

    !call calculate_area
    call calculate_area(radio, area)

    !print result
    write(*,*) 'Area of the circle is:', area

contains

    subroutine calculate_area(radio, area)

        !deactivate implicit typing
        implicit none

        !define variables
        real, intent(in) :: radio
        real, intent(out) :: area

        !calculate area
        area = 4*atan(1.)*radio**2

    end subroutine calculate_area
```

```
51
52 end program circle_area
```

Descrição:

2 Exercício 2

Tarefa: Testar overflow e underflow em precisão simples e dupla.

Código Escrito:

```
!-------------------------------------------------------------
! File: L1-5255417-ex-2.f90
!
! Description:
!   Computes overflow and underflow
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!-------------------------------------------------------------
program testing_overflow_and_underflow

    implicit none

    real test_overflow, real_overflow, test_underflow,
   real_underflow

    real*8 test_overflow8, real_overflow8, test_underflow8,
   real_underflow8

    test_overflow = 1.0
    test_overflow8 = 1.0
    call compute_overflow(test_overflow,real_overflow)
    call compute_overflow8(test_overflow8,real_overflow8)

    test_underflow = 1.0
    test_underflow8 = 1.0
    call compute_underflow(test_underflow,real_underflow)
    call compute_underflow8(test_underflow8,real_underflow8)

    write(*,*)'Overflow simple real order is: ', real_overflow
    write(*,*)'Overflow double real order is: ', real_overflow8

    write(*,*)'Underflow simple real order is: ', real_underflow
    write(*,*)'Underflow double real order is: ', real_underflow8

contains

```

```
42      subroutine compute_overflow(test_overflow,real_overflow)
43
44          implicit none
45
46          real, intent(inout) :: test_overflow
47          real, intent(inout) :: real_overflow
48
49          infinity = huge(1.0_real(4))
50          do while (test_overflow < infinity)
51              write(*,*) test_overflow
52              test_overflow = test_overflow*10
53          end do
54
55      end subroutine compute_overflow
56
57      subroutine compute_overflow8(test_overflow8,real_overflow8)
58
59          implicit none
60
61          real*8, intent(inout) :: test_overflow8
62          real*8, intent(inout) :: real_overflow8
63          real*8 :: infinity
64
65          infinity = huge(1.0_real(8))
66          do while (test_overflow8 < infinity)
67              real_overflow8 = test_overflow8
68              write(*,*) test_overflow8
69              test_overflow8 = test_overflow8*10
70          end do
71
72      end subroutine compute_overflow8
73
74      subroutine compute_underflow(test_underflow,real_underflow)
75
76          implicit none
77
78          real, intent(inout) :: test_underflow
79          real, intent(inout) :: real_underflow
80
81          do while (test_underflow > 3.0E-38)
82              real_underflow = test_underflow
83              write(*,*) test_underflow
84              test_underflow = test_underflow/10
85          end do
86
87      end subroutine compute_underflow
88
89      subroutine compute_underflow8(test_underflow8,real_underflow8)
90
91          implicit none
92
93          real*8, intent(inout) :: test_underflow8
94          real*8, intent(inout) :: real_underflow8
95
96          do while (test_underflow8 > 3.0E-38)
97              real_underflow8 = test_underflow8
98              write(*,*) test_underflow8
99              test_underflow8 = test_underflow8/10
```

```
100         end do
101
102     end subroutine compute_underflow8
103
104 end program testing_overflow_and_underflow
```

Descrição:

3 Exercício 3

Tarefa: Achar a precisão do computador, i.e., o maior número positivo tal que $1 + \epsilon = 1$, usando precisão simples e dupla.

Código Escrito:

```
1  !-------------------------------------------------------------
2  ! File: L1-5255417-ex-3.f90
3  !
4  ! Description:
5  !   Finds computer precision
6  !
7  ! Dependencies:
8  !   - None
9  !
10 ! Since:
11 !   - 03/2025
12 !
13 ! Authors:
14 !   - Pedro C. Delbem <pedrodelbem@usp.br>
15 !-------------------------------------------------------------
16 program computer_precision
17
18     !deactivate implicit typing
19     implicit none
20
21     !define variables
22     real*4 real4, sum4, aux4
23     real*8 real8, sum8, aux8
24
25     !initialize variables
26     real4 = 1.0
27     sum4 = 1.0
28     aux4 = 1.0
29     real8 = 1.0
30     sum8 = 1.0
31     aux8 = 1.0
32
33     !find machine precision
34     do while (sum4+real4 /= sum4) !while sum4+real4 is not equal to
       sum4
35         !divide real4 by 2
36         real4 = aux4
37         aux4 = real4*0.5
```

```
    end do

    !find machine precision
    do while (sum8+real8 /= sum8) !while sum4+real4 is not equal to
    sum4
        !divide real4 by 2
        real8 = aux8
        aux8 = real8*0.5d0
    end do

    !print results
    write(*,*) "Machine precision for real4 is:", real4
    write(*,*) "Machine precision for real8 is:", real8

end program computer_precision
```

Descrição:

4 Exercício 4

Tarefa: Calcular

$$e^{-x} = 1 - x + x^2/2! - x^3/3! + ... \tag{1}$$

para x = 0.1, 1, 10, 100 e 1000 com um erro menor do que 10^{-8}. Problema: quando truncar a série? É preciso calcular o fatorial explicitamente? Comparar o valor obtido usando a série com o resultado exato.

Código Escrito:

```
!-------------------------------------------------------------
! File: L1-5255417-ex-4.f90
!
! Description:
!   Computes taylor series for exponential of -x
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!-------------------------------------------------------------
program exponential_taylor_series

    !deactivate implicit typing
    implicit none

    !define variables
```

```
    integer i
    real*16 x(5), e_x, index, next_term, precision

    !initialize variables
    x = [0.1,1.0,10.0,100.0,1000.0]
    precision = 1.0e-8

    do i=1,5

        !initialize variables
        e_x = 1.0
        index = 1.0
        next_term = 1.0

        !call compute_exponential
        call compute_exponential(x(i), e_x, index, next_term,
   precision)

        !print result
        write(*,*) "    x = ", x(i)
        write(*,*) "    Computed e^(-", x(i), ") = ", e_x
        write(*,*) "    Real e^(-", x(i), ") = ", exp(-x(i))

    end do

contains

    subroutine compute_exponential(x, e_x, index, next_term,
   precision)

        !deactivate implicit typing
        implicit none

        !define variables
        real*16, intent(in) :: x
        real*16, intent(in) :: precision
        real*16, intent(inout) :: index
        real*16, intent(inout) :: next_term
        real*16, intent(out) :: e_x

        !update e_x while the absolute value of next_term is
   greater than precision
        do while (abs(next_term) > precision)

            !compute next term
            next_term = next_term*(-x)/index

            !update e_x
            e_x = e_x + next_term

            !update index and factorial
            index = index + 1

        end do

    end subroutine compute_exponential

end program exponential_taylor_series
```

Descrição:

5 Exercício 5

Tarefa: Considerar a somatória

$$\Sigma(N) = \sum_{n=1}^{2N} (-1)^n \frac{n}{n+1} = -\sum_{n=1}^{N} \frac{2n-1}{2n} + \sum_{n=1}^{N} \frac{2n}{2n+1} = \sum_{n=1}^{N} \frac{1}{2n(2n+1)} \qquad (2)$$

e calcular $\Sigma(N)$ para N = 1, 2, . . . , 10^6 usando as três fórmulas acima. Comparar os resultados usando precis ão simples.

Código Escrito:

```
!-----------------------------------------------------------------
! File: L1-5255417-ex-5.f90
!
! Description:
!   Computes sums of series
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!-----------------------------------------------------------------
program exponential_taylor_series

    !deactivate implicit typing
    implicit none

    !define variables
    integer N(14), i
    real sum1, sum2, sum3

    !initialize variables
    N = [1,2,3,4,5,6,7,9,10**1,10**2,10**3,10**4,10**5,10**6]

    do i=1,14

        !initialize variables
        sum1 = 0.0
        sum2 = 0.0
        sum3 = 0.0

        !compute series
        call compute_series(sum1, sum2, sum3, N(i))
```

```
37
38          !print result
39          write(*,*) N(i), sum1, sum2, sum3
40
41      end do
42
43  contains
44
45      subroutine compute_series(sum1, sum2, sum3, N)
46
47          !deactivate implicit typing
48          implicit none
49
50          !define variables
51          integer i
52          integer, intent(in) :: N
53          real x, sum2a, sum2b
54          real, intent(inout) :: sum1
55          real, intent(inout) :: sum2
56          real, intent(inout) :: sum3
57
58          !initialize partial sums
59          sum2a = 0.0
60          sum2b = 0.0
61
62          !compute sum1
63          do i=1,N
64
65              !update x
66              x = i
67
68              !compute sums
69              sum1 = sum1 + (-1)**(x)*x/(x+1)
70              sum2a = sum2a + (2.0*x-1)/(2.0*x)
71              sum2b = sum2b + (2.0*x)/(2.0*x+1)
72              sum3 = sum3 + 1/(2.0*x*(2.0*x+1))
73
74          end do
75
76          !!! first serie is actually 1 to 2N?  !!!
77
78          !!update sum1
79          !do i=N,2*N
80
81              !!update x
82              !x = i
83
84              !!compute sums
85              !sum1 = sum1 + (-1)**(x)*x/(x+1)
86
87          !end do
88
89          !update sum2
90          sum2 = -sum2a + sum2b
91
92      end subroutine compute_series
93
94  end program exponential_taylor_series
```

Descrição:

6 Exercício 6

Tarefa: Estudar numericamente o erro da aproximação

$$e^{-x} \approx \sum_{n=0}^{N} \frac{(-x)^n}{n!} \tag{3}$$

em função de N, para diferentes valores de x. Sugestão: faça um gráfico do erro em função de N. O que acontece quando e^{-x} é calculado usando a série $e^x = \sum_{n=0}^{N} \frac{x^n}{n!}$ e, depois, calculando $1/e^x$?

Código Escrito:

```
!------------------------------------------------------------
! File: L1-5255417-ex-6.f90
!
! Description:
!   Computes taylor series for exponential of -x with function of N
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!------------------------------------------------------------
program exponential_taylor_series

    !deactivate implicit typing
    implicit none

    !define variables
    integer N(14), i, j
    real exponential_of_minus_x, exponential_of_x, x(5)

    !initialize variables
    N = [1,2,3,4,5,6,7,9,10**1,10**2,10**3,10**4,10**5,10**6]
    x = [0.1,1.0,10.0,100.0,1000.0]

    open(unit=1, file='exponential_taylor_series.txt', status='
   replace')

    do i=1,5

        !initialize variables
        exponential_of_minus_x = 0.0
        exponential_of_x = 0.0
```

```
36
37          write(1,*) 'x:', x(i)
38
39          do j=1,14
40              !compute series
41              call compute_series(exponential_of_minus_x,
    exponential_of_x, N(j), x(i))
42
43              !compute exponential of -x
44              exponential_of_x = 1/exponential_of_x
45
46              !print result
47              write(1,*) 'N:', N(j), exponential_of_minus_x,
    exponential_of_x, exp(-x(i))
48
49          end do
50
51      end do
52
53      close(1)
54
55  contains
56
57      subroutine compute_series(exponential_of_minus_x,
    exponential_of_x, N, x)
58
59          !deactivate implicit typing
60          implicit none
61
62          !define variables
63          integer i
64          integer, intent(in) :: N
65          real j, next_term_minus_x, next_term_x
66          real, intent(in) :: x
67          real, intent(inout) :: exponential_of_minus_x
68          real, intent(inout) :: exponential_of_x
69
70          !initialize next terms
71          next_term_minus_x = 1.0
72          next_term_x = 1.0
73
74          !compute sum1
75          do i=1,N
76
77              !update j
78              j = i
79
80              !compute next terms
81              next_term_minus_x = next_term_minus_x*(-x)/j
82              next_term_x = next_term_x*x/j
83
84              !compute sums
85              exponential_of_minus_x = exponential_of_minus_x +
    next_term_minus_x
86              exponential_of_x = exponential_of_x + next_term_x
87
88          end do
89
```

```
90      end subroutine compute_series
91
92  end program exponential_taylor_series
```

Descrição: