7600017 - Introdução à Física Computacional - Projeto 1

Raphael Vieira Moreira da Serra

14 de Agosto de 2025

Primeira Tarefa

Para calcular os juros nessa tarefa, usei a fórmula:

$$V = \frac{Q \cdot AJM}{N}$$

A sua implementação em Fortran foi da seguinte forma:

```
      PROGRAM JUROS

      WRITE(*,*) 'Valor a ser recebido'
      READ(*,*) Q
      WRITE(*,*) 'Quantidade de parcelas'
      READ(*,*) C
      WRITE(*,*) 'Juros'
      READ(*,*) AJM

      V = (Q/C)*AJM

      WRITE(*,*) V

      END
```

Listing 1: Tarefa 01

O valor retornado para o caso pedido foi de $\approx 183,33$.

Segunda Tarefa

Para realizar essa tarefa, usei as fórmulas de área e volume presentes no formulário:

$$A = 4\pi^2 Rr$$

$$V = 2\pi^2 Rr^2$$

Implementadas no Fortran da seguinte forma:

```
      PROGRAM TORO

      PI = 3.14159

      WRITE(*,*)"Insira o raio interno"
      READ(*,*) R1
      WRITE(*,*)"Insira o raio externo"
      READ(*,*) R2

      AREA = 4 * PI * PI * R1 * R2
      VOL = 2 * PI * PI * R2 * R1 * R1

      WRITE(*,*)"A  REA    : ", AREA
      WRITE(*,*)"O VOLUME   : ", VOL

      END
```

Listing 2: Tarefa 02

Terceira Tarefa

Realizei essa tarefa em 3 passos: *Leitura, Vetorização e Ordenação*. Na etapa de *Leitura*, usei um *loop* para ler o arquivo linha por linha e contar quantas linhas haviam. Na *Vetorização* usei um loop para passar todas as linhas do arquivo para um só vetor afim de facilitar sua manipulação.

Já na etapa de *Ordenação*, usei o algoritmo *Selection Sort* para ordenar o vetor criado na etapa de *Vetorização*. O algoritmo, em PORTUGOL é:

```
para i = 1 até n:
  para j = 1 até n:
    se a[i] > a[j]:
      troque(a[i], a[j])
```

A implementação em Fortran foi da seguinte forma:

```
      PROGRAM ORDEM

      DIMENSION A(100000)

      !Arquivo de entrada
      OPEN(UNIT=1, FILE='tarefa-3-entrada-1.in', status='old')

      !Leitura
      N = 0
10    READ(1,*,END=13) temp
        N = N + 1
        GOTO 10
13    REWIND(1)
      WRITE(*,*) 'N    igual a: ', N

      !Cria   o do vetor
      i = 0
19    READ(1,*,END=23) tem
```

```
19          i = i+1
20          A(i) = tem
21          GOTO 19
22  23    REWIND(1)
23
24        !Ordena  o
25        DO j = 1, N
26          DO k = (j+1), N
27            IF (A(j) .GT. A(k)) THEN
28              sub = A(j)
29              A(j) = A(k)
30              A(k) = sub
31            END IF
32          END DO
33        END DO
34
35        CLOSE(1)
36
37        !Recebimento de M
38        WRITE(*,*) 'Digite uma quantidade de valores menor que N'
39        READ(*,*)M
40
41        !Display dos resultados
42        WRITE(*,*) A(1:M)
43
44        END
```

Listing 3: Tarefa 03

Quarta Tarefa

Para realizar o item A, usei um *loop* com *while* para calcular termos da série até que a diferença entre o valor da série e a da função nativa fossem menores que $\epsilon = 10^{-5}$. Implementado da seguinte forma:

```
1         PROGRAM APPROX
2
3         I = 1 !Contar os termos da s rie
4         FLOG = 0
5         EPS = 1
6
7         WRITE(*,*)"Valor de x a ser calculado"
8         READ(*,*) X
9
10        DO WHILE (EPS .GT. 0.00001)
11          TERMO = ((1 - X)**I)/I
12          FLOG = FLOG - TERMO
13          EPS = ABS(LOG(X) - FLOG)
14          I = I + 1
15        END DO
16
17        WRITE(*,*)'O valor pelo Fortran nativo', LOG(X)
18        WRITE(*,*)'O valor pela s rie', FLOG
19
```

```
20      END
```

Listing 4: Tarefa 04 item A

Para realizar o item B, mudei as variáveis para dupla precisão e permiti que o usuário decidisse o erro da série. Fazendo isso, consegui valores de ϵ da ordem de 10^{-17}.

Segue a implementação em Fortran:

```
1       PROGRAM APPROX
2       IMPLICIT REAL*8(A-H, O-Z)
3
4       I = 1 !Contar os termos da s rie
5       FLOG = 0
6       EPS = 1
7
8       WRITE(*,*)"Valor de x a ser calculado"
9       READ(*,*) X
10
11      WRITE(*,*)"Escolha o valor de Epsilon"
12      READ(*,*) EPSILON
13
14      DO WHILE (EPS .GE. EPSILON)
15        TERMO = ((1 - X)**I)/I
16        FLOG = FLOG - TERMO
17        EPS = ABS(DLOG(X) - FLOG)
18        I = I + 1
19      END DO
20
21      WRITE(*,*)'O valor pelo Fortran nativo', DLOG(X)
22      WRITE(*,*)'O valor pela s rie', FLOG
23      WRITE(*,*)'O valor de epsilon', EPS
24
25      END
```

Listing 5: Tarefa 04 item B

Quinta Tarefa

Para calcular a permutação $N + 1$ a partir de um arquivo contendo a permutação N, usei um algoritmo que atuava da seguinte forma: Para cada linha do arquivo, substituir um dos números e colocar esse número no final da linha. Posteriormente, calculei a paridade. O código FORTRAN foi o seguinte:

```
1       INTEGER FUNCTION FATORIAL(m)
2               FATORIAL = 1
3               DO i = 1, m
4                 FATORIAL = FATORIAL * i
5               END DO
6       END
7
8       PROGRAM PERMUTACAO
9       INTEGER MATRIX(720,721), NEW_MATRIX(720,721)
10      INTEGER FATORIAL
```

```
11      INTEGER arquivo, linhas, colunas, numeros, new_linhas,
    new_colunas
12      arquivo = 1
13
14      !Leitura e Escrita dos valores do arquivo externo numa matriz
15      WRITE(*,*)'Diga a quantidade de n meros que voc  permutou:
    '
16      READ(*,*) numeros
17      linhas = FATORIAL(numeros)
18      colunas = numeros + 1
19
20      OPEN(UNIT=arquivo, FILE='4.txt', STATUS='OLD')
21
22      DO i=1,linhas
23        READ(arquivo,*) (MATRIX(i,j), j=1, colunas)
24      END DO
25
26      CLOSE(arquivo)
27      !Fim da Leitura e Escrita
28
29      !Cria  o de nova matriz
30      new_linhas = FATORIAL(numeros + 1)
31      new_colunas = numeros + 2
32      !Para cada linha, eu crio N c pias na Matriz de Sa da
33      DO i=1, linhas
34        DO j = 0, numeros
35          DO k = 1, numeros+1
36            NEW_MATRIX((4*i) - j, k) = MATRIX(i, k)
37          END DO
38        END DO
39      END DO
40
41      !Come ar a substituir
42      DO i=1, new_linhas, numeros+1
43        DO j=0, numeros
44          IF (j+1 .EQ. colunas) THEN
45                  NEW_MATRIX(i+j, j+1) = numeros+1
46          ELSE
47          k = NEW_MATRIX(i+j, j+1)
48          NEW_MATRIX(i+j, j+1) = numeros+1
49          NEW_MATRIX(i+j, colunas) = k
50          END IF
51        END DO
52      END DO
53
54      !Checando a paridade de cada vetor
55      DO i=1, new_linhas
56        M = 0
57        DO j=1,colunas
58          DO k=j + 1, colunas
59            IF(NEW_MATRIX(i,j) .GT. NEW_MATRIX(i,k)) THEN
60                    M = M + 1
61            ELSE
62            END IF
63          END DO
64        END DO
65      IF (MOD(M,2) .EQ. 0) THEN
```

```
66          NEW_MATRIX(i, new_colunas) = 1
67        ELSE
68          NEW_MATRIX(i, new_colunas) = -1
69        END IF
70        END DO
71
72        !Fim da cria  o
73
74        !Escrevendo num arquivo
75        DO i=1, linhas
76          WRITE(*,*) 'Vetor', i
77          DO j=1, colunas
78            WRITE(*,*)MATRIX(i,j)
79          END DO
80        END DO
81
82        WRITE(*,*) 'Divis o'
83
84        DO i=1, new_linhas
85          WRITE(*,*) 'Vetor ', i
86          DO j=1, new_colunas
87            WRITE(*,*)NEW_MATRIX(i,j)
88          END DO
89        END DO
90
91        !Escrevendo em um arquivo
92        OPEN(UNIT=20, FILE='5.txt', STATUS='REPLACE')
93
94        DO i = 1, new_linhas
95          DO j = 1, numeros + 1
96            WRITE(20, '(I2, 1X)', ADVANCE='NO') NEW_MATRIX(i, j)
97          END DO
98          WRITE(20, '(I3)') NEW_MATRIX(i, new_colunas)
99        END DO
100
101       CLOSE(20)
102
103       END
```

Listing 6: Tarefa 05 item a

Com isso, gerei as permutações $N = 3, 4, 5$, necessárias para o item seguinte. Elas seguem:

```
N = 3
    1 2 3 1
    2 3 1 1
    3 1 2 1
    1 3 2 -1
    2 1 3 -1
    3 2 1 -1

N = 4
1 2 3 4 1
1 2 4 3 -1
```

```
1 3 2 4 -1
1 3 4 2 1
1 4 2 3 1
1 4 3 2 -1
2 1 3 4 -1
2 1 4 3 1
2 3 1 4 1
2 3 4 1 -1
2 4 1 3 -1
2 4 3 1 1
3 1 2 4 1
3 1 4 2 -1
3 2 1 4 -1
3 2 4 1 1
3 4 1 2 1
3 4 2 1 -1
4 1 2 3 1
4 1 3 2 -1
4 2 1 3 -1
4 2 3 1 1
4 3 1 2 1
4 3 2 1 -1

N = 5
 5  2  3  4  1  -1
 1  5  3  4  2  -1
 1  2  5  4  3  -1
 1  2  4  5  3   1
 1  2  4  3  5  -1
 5  2  4  3  1   1
 1  5  4  3  2   1
 1  3  5  4  2   1
 1  3  2  5  4   1
 1  3  2  4  5  -1
 5  3  2  4  1   1
 1  5  4  2  3  -1
 1  3  5  2  4  -1
 1  3  4  5  2  -1
 1  3  4  2  5   1
 5  4  2  3  1  -1
 1  5  2  3  4  -1
 1  4  5  3  2  -1
 1  4  2  5  3  -1
 1  4  3  2  5  -1
 5  4  3  2  1   1
 1  5  3  2  4   1
```

```
 1  4  5  2  3   1
 2  1  3  5  4   1
 2  1  3  4  5  -1
 5  1  3  4  2   1
 2  5  3  4  1   1
 2  1  5  3  4  -1
 2  1  4  5  3  -1
 2  1  4  3  5   1
 5  1  4  3  2  -1
 2  5  1  4  3  -1
 2  3  5  4  1  -1
 2  3  1  5  4  -1
 2  3  1  4  5   1
 5  3  4  1  2   1
 2  5  4  1  3   1
 2  3  5  1  4   1
 2  3  4  5  1   1
 2  4  1  3  5  -1
 5  4  1  3  2   1
 2  5  1  3  4   1
 2  4  5  3  1   1
 2  4  3  5  1  -1
 2  4  3  1  5   1
!Linhas ocultadas pela legibilidade do documento
```

Para calcular o determinante a partir dessa determinação, usei esses arquivos com permutações como "funções"para calcular o somatório.

A implementação foi assim:

```
      INTEGER FUNCTION FATORIAL(m)
              FATORIAL = 1
              DO i = 1, m
                FATORIAL = FATORIAL * i
              END DO
      END

      PROGRAM PERMUTACAO
      INTEGER MATRIX(720,721), NEW_MATRIX(720,721), TESTE(4,4)
      INTEGER FATORIAL
      INTEGER arquivo, linhas, colunas, numeros, new_linhas,
     new_colunas
      INTEGER soma, termo
      arquivo = 1

      !Leitura e Escrita dos valores do arquivo externo numa matriz
      WRITE(*,*)'Diga a quantidade de n meros que voc  permutou:
     '
      READ(*,*) numeros
      linhas = FATORIAL(numeros)
      colunas = numeros + 1

```

```
      OPEN(UNIT=arquivo, FILE='4.txt', STATUS='OLD')

      DO i=1,linhas
        READ(arquivo,*) (MATRIX(i,j), j=1, colunas)
      END DO

      CLOSE(arquivo)

      !Debuggin
      DO i=1, linhas
        WRITE(*,*) 'Vetor', i
        DO j=1, colunas
          WRITE(*,*)MATRIX(i,j)
        END DO
      END DO

      !Matriz de teste
      DO i = 1, 4
        DO j=1, 4
          READ(*,*) K
          teste(i,j) = K
        END DO
      END DO

      !F rmula de Leibniz
      soma = 0
      DO i = 1, linhas
        termo = MATRIX(i, numeros+1)
        DO j = 1, numeros
          termo = termo * TESTE(j, MATRIX(i,j))
        END DO
        soma = soma + termo
      END DO
      WRITE(*,*) soma
      END
```

Listing 7: Tarefa 05

Sexta Tarefa

Para essa tarefa, usei o Método de Monte Carlo para descobrir a região ocupada pelas n-bolas. Gerei $d \cdot M$ números aleatórios, sendo d a quantidade de dimensões a ser calculada e M a quantidade de vezes para rodar a simulação. A fim de descobrir quantos desses pontos estavam dentro da n-bola, usei $\sqrt{x_1^2 + x_2^2 + x_3^2 + ...x_d^2} < 1$.

Segue a implementação em Fortran:

```
      REAL FUNCTION GAMMA(d)
              IF ((d - INT(d)) .NE. 0) THEN
                      GAMMA = GAMMA_HALF_INT(d)
              ELSE
                      gamma = GAMMA_INT(d)
              END IF
              RETURN
```

```
8       END
9
10      REAL FUNCTION GAMMA_INT(d)
11              REAL d, pi
12              PARAMETER (pi = 3.14159265358979323846)
13
14              n = d - 1
15              GAMMA_INT = 1
16              DO i = 1, n
17                GAMMA_INT = GAMMA_INT * i
18              END DO
19      END
20
21      REAL FUNCTION GAMMA_HALF_INT(d)
22              REAL d, pi
23              PARAMETER (pi = 3.14159265358979323846)
24
25              n = INT(d) / 2
26
27              IF (n .EQ. 0) THEN
28                GAMMA_HALF_INT = SQRT(pi) / 2.0
29              ELSE
30                GAMMA_HALF_INT = 2.0
31                DO i = 1, n
32                  GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
33                END DO
34                GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
    n)
35              ENDIF
36              RETURN
37      END
38
39      PROGRAM NBOLAS
40      REAL values(1000), regiao, formula,pi
41      INTEGER dimensions, pontos_dentro, M
42      PARAMETER (pi = 3.14159265358979323846)
43
44      WRITE(*,*)'Insira a quantidade de dimens es desejada: '
45      READ(*,*) dimensions
46      WRITE(*,*) 'Insira M, a quantidade de vezes que vamos simular
    : '
47      READ(*,*) M
48
49      pontos_dentro = 0
50      DO i = 1, M
51        soma = 0
52        DO j = 1, dimensions
53          values(j) = rand()
54          soma = soma + (values(j) * values(j))
55        END DO
56        IF (SQRT(soma) .LT. 1) THEN
57          pontos_dentro = pontos_dentro + 1
58        ELSE
59        ENDIF
60      END DO
61
62      regiao = (2.0**REAL(dimensions)) * (REAL(pontos_dentro)/REAL(
```

```
    M))
63      formula = SQRT(pi**REAL(dimensions)) / GAMMA(1.0  + (REAL(
    dimensio
64     &ns)/2))
65      epsilon = ABS(regiao - formula)
66
67      WRITE(*,*) 'O valor da sua regi o pelo Metodo de Monte Carlo
         :'
68      WRITE(*,*) regiao
69      WRITE(*,*) 'O valor da sua regi o pela f rmula   :'
70      WRITE(*,*) formula
71      WRITE(*,*) 'O erro do seu valor de M   :'
72      WRITE(*,*) epsilon
73      END PROGRAM
```

Listing 8: Tarefa 06

Como uma prova de conceito, calculei o valor de π para $M = 100000000$.

```
 Insira a quantidade de dimensões desejada:
2
 Insira M, a quantidade de vezes que vamos simular:
100000000
 O valor da sua região pelo Metodo de Monte Carlo é:
  3.14143872
 O valor da sua região pela fórmula é:
  3.14159274
 O erro do seu valor de M é:
  1.54018402E-04
```

Sétima Tarefa

Para o primeiro item, usei a fórmula programada no Sexto Item. Segue a implementação:

```
1       REAL FUNCTION GAMMA(d)
2               IF ((d - INT(d)) .NE. 0) THEN
3                       GAMMA = GAMMA_HALF_INT(d)
4               ELSE
5                       gamma = GAMMA_INT(d)
6               END IF
7               RETURN
8       END
9
10      REAL FUNCTION GAMMA_INT(d)
11              REAL d, pi
12              PARAMETER (pi = 3.14159265358979323846)
13
14              n = d - 1
15              GAMMA_INT = 1
16              DO i = 1, n
17                GAMMA_INT = GAMMA_INT * i
```

```
              END DO
      END

      REAL FUNCTION GAMMA_HALF_INT(d)
              REAL d, pi
              PARAMETER (pi = 3.14159265358979323846)

              n = INT(d) / 2

              IF (n .EQ. 0) THEN
                GAMMA_HALF_INT = SQRT(pi) / 2.0
              ELSE
                GAMMA_HALF_INT = 2.0
                DO i = 1, n
                  GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
                END DO
                GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
    n)
              ENDIF
              RETURN
      END

      PROGRAM VOLUMES
      REAL raio, pi
      INTEGER ARQUIVO, dimensions
      PARAMETER (pi = 3.14159265358979323846)
      arquivo = 10
      OPEN(UNIT=arquivo, FILE='saida.txt', STATUS='NEW')

      WRITE(*,*) 'Insira o raio que voc  deseja calcular:'
      READ(*,*) raio
      WRITE(*,*) 'Insira a quantidade de dimens es:'
      READ(*,*) dimensions

      DO i = 0, dimensions
        volume = (raio ** REAL(i)) * (SQRT(pi**REAL(i))/GAMMA(1.0
     & + (REAL(i)/2)))
        WRITE(arquivo, *) volume
      END DO

      CLOSE(UNIT=arquivo)
      END
```

Listing 9: Tarefa 07 item a

Para o segundo item, adaptei o código do primeiro item para escrever em 3 arquivos diferentes e depois plotar o resultado usando *XMGRACE*. Segue o gráfico gerado e o código:

```
      REAL FUNCTION GAMMA(d)
              IF ((d - INT(d)) .NE. 0) THEN
                      GAMMA = GAMMA_HALF_INT(d)
              ELSE
                      gamma = GAMMA_INT(d)
              END IF
              RETURN
      END

```

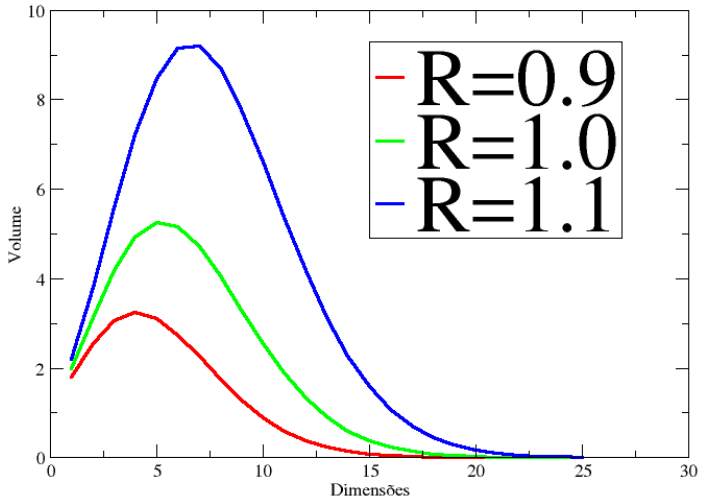

Figura 1: Gráfico para 30 dimensões de n-bolas de raios 0.9, 1.0 e 1.1.

```
10      REAL FUNCTION GAMMA_INT(d)
11              REAL d, pi
12              PARAMETER (pi = 3.14159265358979323846)
13
14              n = d - 1
15              GAMMA_INT = 1
16              DO i = 1, n
17                GAMMA_INT = GAMMA_INT * i
18              END DO
19      END
20
21      REAL FUNCTION GAMMA_HALF_INT(d)
22              REAL d, pi
23              PARAMETER (pi = 3.14159265358979323846)
24
25              n = INT(d) / 2
26
27              IF (n .EQ. 0) THEN
28                GAMMA_HALF_INT = SQRT(pi) / 2.0
29              ELSE
30                GAMMA_HALF_INT = 2.0
31                DO i = 1, n
32                  GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
33                END DO
34                GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
    n)
35              ENDIF
```

```
36              RETURN
37      END
38
39      REAL FUNCTION VOLUME(raio, d)
40              REAL raio
41              INTEGER d
42              PARAMETER (pi = 3.14159265358979323846)
43
44              VOLUME = (raio ** REAL(d)) * (SQRT(pi**REAL(d))/GAMMA
    (1.0
45     & + (REAL(d)/2)))
46
47      END
48
49      PROGRAM VOLUMES
50      REAL raios(3), pi
51      INTEGER arquivo_1, arquivo_2, arquivo_3, dimensions
52      PARAMETER (pi = 3.14159265358979323846)
53      raios(1) = 0.9
54      raios(2) = 1.0
55      raios(3) = 1.1
56      arquivo_1 = 1
57      arquivo_2 = 2
58      arquivo_3 = 3
59      OPEN(UNIT=arquivo_1, FILE='dados1.dat', STATUS='UNKNOWN')
60      OPEN(UNIT=arquivo_2, FILE='dados2.dat', STATUS='UNKNOWN')
61      OPEN(UNIT=arquivo_3, FILE='dados3.dat', STATUS='UNKNOWN')
62
63      DO i = 1,3
64        DO j = 1, 25
65          x = REAL(j)
66          WRITE(i, *) x, VOLUME(raios(i), j)
67        END DO
68      END DO
69
70      CLOSE(UNIT=1)
71      CLOSE(UNIT=2)
72      CLOSE(UNIT=3)
73      END
```

Listing 10: Tarefa 07 item b

Oitava Tarefa

Para calcular, reaproveitei o código da Sétima Questão e o re-adaptei para calcular a razão entre a n-bola e um n-cubo.

```
1       REAL FUNCTION GAMMA(d)
2               IF ((d - INT(d)) .NE. 0) THEN
3                       GAMMA = GAMMA_HALF_INT(d)
4               ELSE
5                       gamma = GAMMA_INT(d)
6               END IF
7               RETURN
8       END
```

```
9
10      REAL FUNCTION GAMMA_INT(d)
11              REAL d, pi
12              PARAMETER (pi = 3.14159265358979323846)
13
14              n = d - 1
15              GAMMA_INT = 1
16              DO i = 1, n
17                GAMMA_INT = GAMMA_INT * i
18              END DO
19      END
20
21      REAL FUNCTION GAMMA_HALF_INT(d)
22              REAL d, pi
23              PARAMETER (pi = 3.14159265358979323846)
24
25              n = INT(d) / 2
26
27              IF (n .EQ. 0) THEN
28                GAMMA_HALF_INT = SQRT(pi) / 2.0
29              ELSE
30                GAMMA_HALF_INT = 2.0
31                DO i = 1, n
32                  GAMMA_HALF_INT = GAMMA_HALF_INT * (2.0 * i - 1.0)
33                END DO
34                GAMMA_HALF_INT = GAMMA_HALF_INT * SQRT(pi) / (2.0**
    n)
35              ENDIF
36              RETURN
37      END
38
39      PROGRAM VOLUMES
40      REAL raio, pi
41      INTEGER ARQUIVO, dimensions
42      PARAMETER (pi = 3.14159265358979323846)
43
44      WRITE(*,*) 'Insira o raio que voc  deseja calcular:'
45      READ(*,*) raio
46      WRITE(*,*) 'Insira a quantidade de dimens es:'
47      READ(*,*) dimensions
48
49      DO i = 1, dimensions
50        volume = (raio ** REAL(i)) * (SQRT(pi**REAL(i))/GAMMA(1.0
51     & + (REAL(i)/2)))
52        WRITE(*,*) (2.0**REAL(i))/volume
53      END DO
54      END
```

Listing 11: Tarefa 08

Para aproximar o limite $d \to \infty$, simulei para 75 dimensões. Segue o output:

```
Insira o raio que você deseja calcular:
1
 Insira a quantidade de dimensões:
75
   1.00000000
```

1.27323949
1.90985906
3.24227762
6.07927036
12.3845882
27.0912876
63.0741882
155.221634
401.542755
1086.98877
3067.56055
8995.97949
27340.1738
85905.2812
278484.719
929712.938
3191199.75
11245600.0
40631612.0
150342592.
569071616.
2.20135424E+09
8.69477274E+09
3.50356439E+10
1.43916876E+11
6.02218299E+11
2.56536897E+12
1.11181373E+13
4.89949359E+13
2.19418806E+14
9.98116711E+14
4.60964935E+15
2.16043040E+16
1.02710793E+17
4.95134172E+17
2.41934559E+18
1.19780654E+19
6.00679255E+19
3.05018881E+20
1.56785756E+21
8.15560403E+21
4.29195476E+22
2.28448838E+23
1.22955461E+24
6.69001150E+24
3.67896630E+25

```
2.04431688E+26
1.14763024E+27
6.50726281E+27
3.72608033E+28
2.15417906E+29
1.25721115E+30
7.40552173E+30
4.40200984E+31
2.64012107E+32
1.59737175E+33
9.74836681E+33
5.99981901E+34
3.72360202E+35
2.32995765E+36
1.46972159E+37
9.34477238E+37
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
      Infinity
```