O principal tema do EP2 foi a quadratura gaussiana para integrais duplas e, nesse sentido, criar um algoritmo em C/C++ ou python capaz de resolver integrais duplas. O problema foi dividido em duas partes: O algoritmo de mudança de variável dos limites de integração, e o algoritmo principal de resolução de integrais simples e duplas pelo método da quadratura gaussiana exposto no enunciado.

A implementação dos algoritmos foi feita em Python 3.10.4 e com o uso das bibliotecas numpy e math e com o auxílio da IDE VScode.

Para a primeira parte cria-se duas funções distintas: a função responsável por devolver os novos valores dos limites de integração e outra para novo valor de “dt”. Nesse sentido implementa-se a função chamada mudança_de_base que recebe os valores de a, b (limites inferior e superior) e t (valor de x). Já a função que devolve o valor de dt tem como parâmetros apenas a e b. Segue o código das funções:

```python
def mudanca_de_base(a,b, t):

    return (a+b)/2 + ((b-a)/2 * t)

def dt (a,b):

    return (b-a)/2
```

Para a segunda do EP cria-se uma função cujo os parâmetros são: n (número de nós e pesos), a (limite inferior), b (limite superior), x (valor da variável x), f (função a ser integrada). Nesse sentido, o algoritmos exposto no enunciado é executado para n valores de nós e pesos cujos valores foram dados e implementado nos vetores abaixo:

```python
global w3
global x3

w3 = np.array([0.888889, 0.555556, 0.555556])
x3 = np.array([0, 0.774597, -0.774597])


global w6
global x6

w6 = np.array([0.4679139345726910473898703, 0.4679139345726910473898703, 0.3607615730481386075698335, 0.3607615730481386075698335, 0.1713244923791703450402961, 0.1713244923791703450402961])
x6 = np.array([0.2386191860831969086305017, -0.2386191860831969086305017, 0.6612093864662645136613996, -0.6612093864662645136613996, 0.9324695142031520278123016, -0.9324695142031520278123016])

global w8
global x8

w8 = np.array([0.3626837833783619829651504, 0.3626837833783619829651504, 0.3137066458778872873379622, 0.3137066458778872873379622, 0.2223810344533744705443560, 0.2223810344533744705443560, 0.1012285362903762591525314, 0.1012285362903762591525314])
x8 = np.array([0.1834346424956498049394761, -0.1834346424956498049394761, 0.5255324099163289858177390, -0.5255324099163289858177390, 0.7966664774136267395915539, -0.7966664774136267395915539, 0.9602898564975362316835609, -0.9602898564975362316835609])

global w10
global x10

w10 = np.array([0.2955242247147528701738930, 0.2955242247147528701738930, 0.2692667193099963550912269, 0.2692667193099963550912269, 0.2190863625159820439955349, 0.2190863625159820439955349, 0.1494513491505805931457763, 0.1494513491505805931457763, 0.0666713443086881375935688, 0.0666713443086881375935688])
x10 = np.array([0.1488743389816312108848260, -0.1488743389816312108848260, 0.4333953941292471907992659, -0.4333953941292471907992659, 0.6794095682990244062343274, -0.6794095682990244062343274, 0.8650633666889845107320967, -0.8650633666889845107320967, 0.9739065285171717200779640, -0.9739065285171717200779640
```

E o código do algoritmo já com a mudança de base implementada abaixo:

```python
def quadratura_gaussiana(n, a, b, x,f):

    integral = 0

    if int(n) != 6 and int(n) != 8 and int(n) != 10:
        return 0


    elif n == 6:
        for i in range (n):
            integral = integral + F(x,mudanca_de_base(float(a),float(b), x6[i]),f)*w6[i]
    elif n == 8:
        for i in range (n):
            integral = integral + F(x,mudanca_de_base(float(a),float(b), x8[i]),f)*w8[i]
    elif n == 10:
        for i in range (n):
            integral = integral + F(x,mudanca_de_base(float(a),float(b), x10[i]),f)*w10[i]

    return integral
```

Note que no final do algoritmo não multiplica-se o resultado final pelo valor de dt pois por se tratar de integrais duplas dt é inserido na segunda função quadratura_gaussiana_2d cujo código é parecido com a função anterior. Onde soma-se os valores devolvidos pela a função anterior (quadratura_gaussiana) porém com parâmetros de a e b em função de y após o método de mudança de variável. segue o código:

```
def quadratura_gaussiana_2d(n, a, b,f):

    integral = 0
    if int(n) != 6 and int(n) != 8 and int(n) != 10:
        return 0
    elif n == 6:
        for i in range (n):
            integral = integral + quadratura_gaussiana(int(n), Fa(mudanca_de_base(float(a),float(b), x6[i])), Fb(mudanca_de_base(float(a),float(b), x6[i])), mudanca_de_base(float(a),float(b), x6[i]),f)*(dt(Fa(mudanca_de_base(float(a),float(b), x6[i])),Fb(mudanca_de_base(float(a),float(b), x6[i]))))*w6[i]
    elif n == 8:
        for i in range (n):
            integral = integral + quadratura_gaussiana(int(n), Fa(mudanca_de_base(float(a),float(b), x8[i])), Fb(mudanca_de_base(float(a),float(b), x8[i])), mudanca_de_base(float(a),float(b), x8[i]),f)*(dt(Fa(mudanca_de_base(float(a),float(b), x8[i])),Fb(mudanca_de_base(float(a),float(b), x8[i]))))*w8[i]
    elif n == 10:
        for i in range (n):
            integral = integral + quadratura_gaussiana(int(n), Fa(mudanca_de_base(float(a),float(b), x10[i])), Fb(mudanca_de_base(float(a),float(b), x10[i])), mudanca_de_base(float(a),float(b), x10[i]),f)*(dt(Fa(mudanca_de_base(float(a),float(b), x10[i])),Fb(mudanca_de_base(float(a),float(b), x10[i]))))*w10

    return integral * dt(float(a),float(b))
```

Na resolução dos exemplos enunciados foram usados os seguintes parâmetros:

Ex1:

Para cálculo do volume cubo:

Função = 1
limite inferior de x = 0
limite superior de x = 1
limite inferior de y = 0
limite superior de y = 1

$$\int_0^1 \int_0^1 1 \, dx \, dy$$

resultado exato: 1
resultado numérico para n:

```
Volume do cubo para n = 6:   1.0
Volume do cubo para  n = 8:   1.0
Volume do cubo para n = 10:   1.0
```

Para o cálculo do volume do tetraedro:

Função = -x - y + 1
limite inferior de x = 0
limite superior de x = -y + 1
limite inferior de y = 0
limite superior de y = 1

$$\int_0^1 \int_0^{-y+1} (-x - y + 1) \, dx \, dy$$

resultado exato: $\frac{1}{6}$

resultado numérico para n:

```
Volume do tetraedro para n = 6:  0.16666666666666666
Volume do tetraedro para n = 8:  0.16666666666666666
Volume do tetraedro para n = 10:  0.16666666666666666
```

Ex2:

Segue o calculo numérico da integral dada:

$$A = \int_0^1 \left[\int_0^{1-x^2} dy \right] dx = \int_0^1 \left[\int_0^{\sqrt{1-y}} dx \right] dy = \frac{2}{3}$$

```
Resulatado de A dydx para n = 6:  0.6666666666666666
Resulatado de A dydx para n = 8:  0.6666666666666667
Resulatado de A dydx para n = 10:  0.6666666666666667
Resulatado de A dxdy para n = 6:  0.6670464379156135
Resulatado de A dxdy para n = 8:  0.6668355801001765
Resulatado de A dxdy para n = 10:  0.6667560429365089
```

Observe que para dxdy o resultado se mostra menos preciso, isso deve muito por conta do fato de aproximar raízes ser mais impreciso que para potências.

Ex3:

Volume da área abaixo da superfície:

Função = $e^{\frac{x}{y}}$

limite inferior de x = y^3

limite superior de x = y^2

limite inferior de y = 0.1

limite superior de y = 0.5

$$\int_{0.1}^{0.5} \int_{y^3}^{y^2} e^{x/y}\, dx\, dy$$

Resultado numérico para n:

```
Volume da regiao abaixo da superficie para n = 6:  0.03330556611623719
Volume da regiao abaixo da superficie para n = 8:  0.03330556611623208
Volume da regiao abaixo da superficie para n = 10:  0.03330556611623208
```

Area da superficie:

Função = $\sqrt{1+\frac{e^{(2x)/y}x^2}{y^4}+\frac{e^{(2x)/y}}{y^2}}$

limite inferior de x = y^3

limite superior de x = y^2

limite inferior de y = 0.1

limite superior de y = 0.5

$$\int_{0.1}^{0.5}\int_{y^3}^{y^2}\sqrt{1+\frac{e^{(2x)/y}x^2}{y^4}+\frac{e^{(2x)/y}}{y^2}}\,dx\,dy$$

Resultado numérico para n:

```
Area da superficie para n = 6:  0.10549788240049787
Area da superficie para n = 8:  0.10549788240051997
Area da superficie para n = 10:  0.10549788240051992
```

Ex4:

Para o volume da calota:

Função = $x2\pi$

limite inferior de = 0

limite superior de x = $\sqrt{1-\left(y+\frac{3}{4}\right)^2}$

limite inferior de y = 0

limite superior de y = $\frac{1}{4}$

$$2\pi\int_0^{\frac{1}{4}}\int_0^{\sqrt{1-\left(y+\frac{3}{4}\right)^2}} x\,dx\,dy$$

Resultado exato: $\frac{11\pi}{192}$

Resultado numérico para n:

```
Volume da calota esferica para n = 6:  0.1799870791119152
Volume da calota esferica para n = 8:  0.17998707911191522
Volume da calota esferica para n = 10:  0.17998707911191525
```

Para o volume do sólido de revolução:

Função = $2\pi x$
limite inferior de x = 0
limite superior de x = $e^{(-y^2)}$
limite inferior de y = -1
limite superior de y = 1

$$\int_{-1}^{1}\int_{0}^{e^{-y^2}} 2\pi x \, dx \, dy$$

Resultado número para n:

```
Volume do sólido de rotacao para n = 6:  3.75816503289671
Volume do sólido de rotacao para n = 8:  3.7582492624394392
Volume do sólido de rotacao para n = 10:  3.7582496332093873
```

Na interface do programa há uma opção também para executar integrais duplas quaisquer dxdy porém devido a forma com que o código foi implementado deve-se inverter x por y na função, por exemplo F(x,y) = $x^2 + y$ deve-se escrever no terminal : y**2 + x.